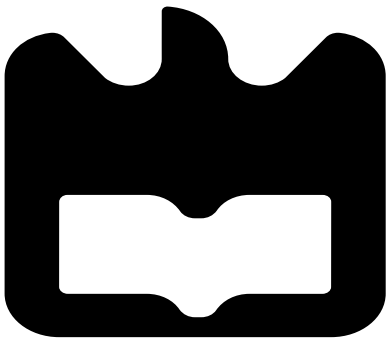
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2010

Diogo Miguel do Céu Condeço

Sistema de áudio referência para ambientes interiores e exteriores



Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2010

Diogo Miguel do Céu Condeço

Sistema de áudio referência para ambientes interiores e exteriores

"Doubt is the father of invention."

— Galileo Galilei

Diogo Miguel do Céu Condeço

Sistema de áudio referência para ambientes interiores e exteriores

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações (Mestrado Integrado), realizada sob a orientação científica do Dr. Nuno Borges de Carvalho, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president

Prof. Dr. Paulo Jorge dos Santos Gonçalves Ferreira
Professor catedrático da Universidade de Aveiro

vogais / examiners committee

Prof. Dr. Fernando José da Silva Velez
Professor auxiliar da Faculdade de Engenharia da Universidade da Beira Interior

Prof. Dr. Nuno Borges Carvalho
Professor associado com agregação da Universidade do Aveiro (Orientador)

agradecimentos / acknowledgements

Aproveito este curto espaço para deixar uma palavra de apreço a todas as pessoas que contribuíram directa ou indirectamente para o resultado final deste projecto.

Ao meu orientador Prof. Dr. Nuno Borges de Carvalho pela oportunidade de desenvolver este projecto, a orientação, motivação e imaginação contagiantes que deposita nos projectos.

Aos meus grandes amigos, como é o caso do Rui, do Prof. João Paulo, do Prof. Paulo Pina entre outros, e ao meu irmão que me influenciaram positivamente e desencadearam o gosto pelo ramo electrónico da engenharia.

Aos meus pais um agradecimento muito especial e sentido pela forma incansável como me apoiaram e possibilitaram esta oportunidade de completar um longo ciclo académico.

Palavras-chave GPS, ZigBee, localização, guia sono, AudioGuide, C

Resumo

Actualmente, as tecnologias de localização são uma área em continua expansão. As aplicações e benefícios destas tecnologias são variadas, mas encontram-se mais focadas na localização de objectos dando pouco relevo à qualidade da informação que transmitem.

Esta dissertação propõe desenvolver um sistema que incorpore meios de localização exteriores e interiores para difusão de informação sonora detalhada. A solução proposta, contempla o desenvolvimento de um protótipo que consiga conjugar o sistema GPS e o sistema ZigBee de forma a analisar referências recolhidas por estes sistemas. Esta informação, permite determinar o ambiente que rodeia o utilizador, e assim seleccionar a informação a difundir. O dispositivo terá a funcionalidade de suportar vários idiomas, e de poder partilhar a informação por rádio FM com outros dispositivos adjacentes que suportem a recepção destes sinais.

Keywords GPS, ZigBee, location, AudioGuide, C

Abstract

Nowadays, location thecniques are in continuous development. There are serveral applications and benefits that we can take from these techniques, but their main goal is location of objects, giving little attention to the quality of information provided.

This thesis proposes to develop a system that incorporates means for both indoor and outdoor location providing detailed audio information. This solution includes the development of a prototype system that can combine GPS and ZigBee in order to analyze refences collected by both systems. This information sets the surrounding environment, and by so selecting the information to be broadcasted. The device has the functionality to support multiple languages, and should be able to share information by FM radio with adjacent users with compatible devices.

Conteúdo

Glossário

A-GPS Assisted Global Positioning System

AAC-LC Low Complexity Advanced Audio Coding

ABR Available Bit Rate

ADPCM Adaptive Differential Pulse Code Modulation

ASCII American Standard Code for Information Interchange

AVR Advanced Virtual Risc

AVR32 32bits Advanced Virtual Risc

CAD Computer-Aided Design

CBR Constant Bit Rate

CD Compact Disc

CS Chip Select

DFU Device Firmware Upgrade

DoD Department of Defence

DRC Design Rule Check(ing)

FM Frequency Modulation

GPS Global Positioning System

I/O Input/Output

I²C Inter-Integrated Circuit

ID Identification

IDE Integrated Development Environment

IEEE Institute of Electrical & Electronics Engineers, Inc.

IMA Interactive Multimedia Association

ISP In-System Programming

JTAG Joint Test Action Group

LOPES Localização de Pessoas

MCU MicroController Unit

MISO Master Input Slave Output

MOSI Master Output Slave Input

MP3 MPEG-1 or MPEG-2 Audio Layer 3 (or III)

MPEG Moving Picture Experts Group

NMEA National Marine Electronics Association

O-TDOA Observed Time Difference of Arrival

Pb-Free Lead Free

PCB Printed Circuit Board

PCM Pulse Code Modulation

PDA Personal Digital Assistant

PDP-11 Programmed Data Processor model 11

POI Points of Interest

RAM Random-Access Memory

RF Radio Frequency

RFID Radio Frequency Identification

RoHS Reduction of Hazardous Substances

RS-232 Recommended Standard 232

SCK Shared Clock

SMD Surface-Mounted Device

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver/Transmitter

USART Universal Synchronous Asynchronous Receiver/Transmitter

USB Universal Serial Bus

VBR Variable Bit Rate

WI-FI Wireless Fidelity

WMA Windows Media Audio

XML Extensible Markup Language

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A localização de pessoas e objectos é um assunto que levanta um elevado interesse junto das comunidades académicas e científicas.

São variados os exemplos que fazem uso da informação das tecnologias de localização e que traduzem bem a importância que esta área de investigação assume globalmente, sendo a sua utilização transversal aos diferentes meios (militar e civil).

Quando se fala de: sistemas de misseis guiados; aviões não tripulados; controlo de tráfego aéreo; sistemas de navegação móvel, entre outros, todos eles tem como base de funcionamento a tecnologia de localização. Todo este interesse gera a necessidade de melhorar os serviços existentes e desenvolver novas funcionalidades nos sistemas, algo que efectivamente se tem verificado a um ritmo elevado ao longo dos últimos anos. Prova disso é a diminuição dos custos e dimensões dos equipamentos e a melhoria na sua precisão.

Os sistemas de localização definem um conjunto de dispositivos, técnicas, algoritmos e aplicações, que estimam a posição absoluta ou relativa de um objecto num determinado ambiente. Como a quantidade de cenários possíveis é muito variado, os sistemas de localização têm que ser diversificados, sendo extremamente difícil satisfazer todas os possibilidades com apenas uma tecnologia.

Os sistemas de localização tiveram a sua origem no meio militar, onde surgiu a necessidade de localizar os diferentes meios nos mais variados ambientes. Por esta razão, e devido à diversidade de situações que podem ocorrer, a infraestrutura que sustenta esta actividade, tem que apresentar uma cobertura geral. Como os meios de transporte da informação são o ar ou o vazio, através do quais se propagam as ondas radio-eléctricas, é necessário criar uma rede de transmissores que garanta o máximo de cobertura possível.

A solução a este problema é a criação de duas redes de estações: uma terrestre; e uma orbital de satélites.

O sistema norte-americano GPS (Global Positioning System) ou o sistema europeu Galileo - ainda em desenvolvimento, são dois bons exemplos da implementação prática desta solução.

As redes celulares possuem também, serviços que permitem a localização através de técnicas como o Cell ID, O-TDOA (Oriented Time Difference in Arrival) e o A-GPS (Assisted GPS).

Enquanto que no caso de localização no exterior (*outdoor*), a ideia de duas redes de transmissores é uma solução viável, na situação em que o objecto se encontrar num ambiente interior tal não se verifica.

Nos ambientes interiores é difícil os sinais dos transmissores penetrarem pelos obstáculos que encontram, nomeadamente telhados e paredes. Para além disso, tratam-se de meios de dimensões reduzidas, muito menores que as distâncias às estações transmissoras, podendo ocorrer erros incomportáveis, de tal forma que os resultados obtidos não façam sentido na prática.

Os primeiros sistemas de localização interiores eram baseados em sistemas infravermelhos, e adoptavam uma localização relativa, ou seja, a posição seguinte dependia da anterior. Inicialmente desenvolvido para ambientes industriais, estes sistemas tornaram-se globais com o desenvolvimento dos computadores, mais propriamente os ratos informáticos. Eram sistemas que utilizavam dois sensores infravermelhos para medirem as deslocações nos dois eixos (X e Y) e a partir desses valores calcularem a nova posição.

Nos últimos anos tem-se verificado uma constante evolução no campo das tecnologias de comunicação sem fios, podendo também estas serem utilizadas para a localização através da medição dos sinais de rádio-frequência (RF), ou da medição do tempo de viagem que a informação demora entre o emissor e o receptor.

Neste campo, tecnologias como: RFID (Radio-Frequency Identification); Wi-Fi (Wireless Fidelity – IEEE 802.11); Bluetooth (IEEE 802.15.1); e ZigBee (IEEE 802.15.4); apresentam-se como tecnologias predominantes devido ao seu baixo custo consequência da sua sucedida globalização.

Por se tratarem de tecnologias com emissores de rádio-frequência, que por natureza são pouco eficientes em termos de energia despendida, a investigação científica continua uma procura contínua por melhorar estas técnicas. Existe assim um trabalho intenso de

pesquisa, quer por novas técnicas, quer pelo melhoramento de eficiência e diminuição de custo das actuais.

Porém as tecnologias baseadas na medição do sinal RF, são bastante instáveis, uma vez que os sinais são afectados por diversos factores, inerentes à natureza do sinal e ao ambiente à sua volta. Num ambiente interior, existem vários objectos (paredes, cadeiras, mesas, pessoas) que perturbam a propagação do sinal, de variadas maneiras (refracção, reflexão, absorção), de tal forma que a estabilidade e potência do sinal é seriamente afectada. Por esta razão, tal como o GPS, há que ter em conta que para cada estimativa calculada, existe sempre um erro associado, e imprevisível.

O GPS pela sua presença predominante no mercado e o ZigBee pelo baixo consumo e flexibilidade funcional foram as tecnologias adoptadas para a obtenção da informação de localização nesta dissertação.

O desenvolvimento do projecto, pressupõe a existência de cobertura GPS e/ou rede ZigBee, baseada na plataforma CC243X da Texas Instruments.

1.1 Enquadramento do Projecto

A realização deste projecto procura dar resposta à necessidade de transmitir informação aos utilizadores de determinadas infraestruturas. Assim, recorrendo a sistemas de localização, pretende-se que as pessoas recebam informação (sonora) com dados que os administradores dessas infraestruturas achem relevantes.

Numa utilização interior, o objectivo será receber informação dos objectos, à medida que o utilizador se aproxima dos mesmos. Esta informação, disponibilizada pelas entidades interessadas, encontra-se previamente carregada no cartão de memória do dispositivo.

Numa escala mais abrangente pretende-se aplicar o mesmo procedimento na rede GPS, ou seja, num ambiente exterior, em que os edifícios, monumentos e outros pontos de interesse, passam a ser os objectos dos quais se pretende transmitir a informação.

Esta ideia surge assim como uma nova funcionalidade que pode ser integrada em redes ZigBee criadas durante projectos anteriores. Podem também, os módulos ZigBee, funcionarem de modo independente, ou seja, como *beacons* que emitem a sua referência sem se encontrarem ligados a nenhuma rede.

1.2 Objectivos

O objectivo geral desta dissertação é desenvolver um sistema de referências sonoras, com recurso às tecnologias GPS e ZigBee. Pretende-se que este sistema seja capaz de reproduzir a informação sonora que possui para as determinadas referências geográficas que surjam.

Os módulos GPS (no exterior) e ZigBee (no interior) recebem informação sobre o ambiente em que se encontram e transmitem-na ao microcontrolador principal. A informação é processada e, são recolhidos os dados sonoros junto do cartão memória que são reproduzidos ao utilizador.

As premissas essenciais deste sistema são: o baixo consumo; o baixo custo; e a portabilidade. Pretende-se assim uma solução final com um custo de produção baixo, que apresente um tempo de utilização elevado, e que o utilizador o possa transportar sem dificuldades.

O protótipo final deve permitir ao utilizador receber os dados de cada elemento, de um forma automática e autónoma, do ambiente que o rodeia, bem como no futuro receber outro tipo de informação que as entidades distribuidoras do serviço entendam apropriadas.

1.3 Estrutura da Tese

A dissertação foi organizada por capítulos com a seguinte estrutura:

O capítulo 2 é uma introdução geral às soluções existentes de localização GPS e ZigBee, que são a ponto de partida para a escolha da informação a fornecer.

No capítulo 3 são explicadas as tecnologias utilizadas pelo dispositivo final, nomeadamente os princípios de funcionamento das mesmas.

No capítulo 4 são apresentados os principais blocos de hardware que constituem a solução final obtida, com destaque para os detalhes relevantes para a sua escolha e as funcionalidades que transmitem para o resultado final.

No capítulo 5 é apresentado o conjunto de software usado no desenvolvimento deste trabalho, bem como a linguagem de programação adoptada.

No capítulo 6 é apresentada e descrita a solução final obtida, com especial relevo aos detalhes do seu funcionamento.

No capítulo 7 são descritas as funcionalidades que o utilizador pode alterar e o procedimento que deve efectuar para tal.

No capítulo 8 são analisados os resultados e desempenho do dispositivo.

No capítulo 9 são apresentadas as conclusões obtidas durante a realização do projecto, bem como a análise de melhorias e novas funcionalidades que podem ser adicionadas ao dispositivo final.

Capítulo 2

Estado de Arte

Actualmente, existem vários sistemas de navegação e orientação, que em geral podem ser divididos em duas categorias: interiores e exteriores.

Sistemas que contemplem os dois meios são pouco comuns, devido ao facto de não existir uma tecnologia que permita satisfazer os dois cenários.

A incompatibilidade entre os dois meios deve-se sobretudo a uma relação cobertura/erros na medição. Enquanto que nos sistemas interiores a precisão é o elemento determinante na tecnologia a escolher em detrimento da cobertura, no exterior o que prevalece é a necessidade por uma cobertura global.

Em todo o caso, ambos os tipos de sistemas estão orientados para uma difusão mais quantitativa da informação, ou seja, a preocupação principal é a navegação e orientação de um objecto em relação a outro(s). A qualidade da informação que é fornecida é negligenciada, por ser pouco importante nesse cenário. Este projecto pretende contrariar esse princípio, dando maior atenção à informação que se pretende dar, nomeadamente prestar um serviço em que o utilizador tenha acesso a informação detalhada sobre o(s) objecto(s) que o rodeia(m).

Os sistemas de navegação e identificação de objectos, tanto para ambientes exteriores como interiores, são de particular interesse para pessoas que possuem deficiências na visão, e surgem como um importante meio de assistência na sua orientação, melhorando significativamente a sua qualidade de vida. No entanto, este tipo de sistemas permitem minimizar outros tipos de barreiras comuns à população em geral, como pode ser o caso do idioma, da falta de conhecimento de navegação num determinado lugar, ou falta de marcos geográficos, entre outras.

2.1 Sistemas de Localização em Ambientes Exteriores

No exterior, o GPS, que tem sofrido constantes melhoramentos, assume-se como a tecnologia por excelência, devido sobretudo a quatro características de eleição para os utilizadores:

- Baixo custo;
- Reduzidas dimensões;
- Cobertura generalizada;
- Relativa boa precisão para os fins a que se destina.

Estes sistemas de localização são desenhados para que o utilizador, além de saber a sua posição, receba informação do meio que o rodeia. Assim sendo, cada utilizador dispõe de um dispositivo portátil onde pode visualizar mapas digitais, por exemplo um PDA, ou um receptor GPS. A posição recebida a partir do GPS é usada para mostrar, no mapa, a localização do respectivo utilizador. A maioria destes dispositivos possui também um periférico de áudio que fornece informação, nomeadamente direcções para o utilizador se deslocar para a localização desejada.

Figura 2.1: Receptor GPS.

Existem alguns serviços que permitem definir pontos de interesse (POI - Points of Interest) que permitem ao utilizador definir alertas visuais e sonoros que são activados pela proximidade desses locais. A Garmin [1], disponibiliza esse serviço nos vários modelos que comercializa que suportem esta norma. O utilizador é livre de criar ou distribuir os seus pontos de interesse, bem como de usar os pontos de terceiros, apesar de se tratar de um formato fechado.

Existem também alguns sistemas de localização e identificação de objectos mais centrados na curta distância em ambientes exterior [2], em que duas utilizações muito comuns são: a identificação de obstáculos para pessoas com deficiências visuais [3] [4]; e a identifica de obstáculos em vias rodoviárias [5].

Grande parte dos sistema de navegação possuem interfaces sonoros que permitem a reprodução de informação, que geralmente se limita a indicações geográficas e a alertas. A constante necessidade dos utilizadores por mais e melhor informação origina com que esta limitação seja cada vez mais explorada. É esta lacuna que este projecto se pressupõe a preencher.

2.2 Sistemas de Localização em Ambientes Interiores

A necessidade de uma elevada precisão nos sistemas de localização interiores é uma condição imposta pela dimensão dos objectos em questão. A utilização deste tipo de sistemas é muito específica, pelo que para a sua cobertura é geralmente estabelecida pela inclusão de um ou mais sensores fixos por divisão dos edifícios em são implementados.

Se o GPS, no exterior, assume-se quase como a tecnologia por defeito a ser utilizada, no meio interior existe um conjunto de tecnologias muito alargado que tentam fazer frente às necessidades do mercado. Algumas destas tecnologias com maior destaque são: RFID [6], ZigBee [7], entre outras.

A tecnologia ZigBee por apresentar uma melhor relação alcance/consumo de energia [7], foi a solução adoptada. Para além disso o objectivo de produzir um circuito com um preço reduzido também veio influenciar a escolha, bem como o acesso mais fácil a este tipo de equipamentos. Em termos práticos o RFID adopta uma topologia em que o receptor de informação se encontra numa posição fixa, sendo que neste caso o que se pretende é exactamente o oposto.

2.2.1 ZigBee

O ZigBee é o nome comum da norma 802.15.4 trata-se de uma rede de baixo custo, de baixa taxa de transferência e de baixo consumo [8]. Trata-se de uma rede *Mesh*, que se auto-estabelece assim que o primeiro dispositivo arranca e não detecta a presença de outro responsável pela gestão da mesma. A comunicação entre pontos, pode ser efectuada quer

directamente - caso se encontrem ao alcance um do outro - quer indirectamente, onde a informação passa por outros pontos que não o de destino nem o de origem.

Os sensores ZigBee, entre outros, são frequentemente utilizados com rádio-faróis, sinalizando a sua presença. Este princípio de funcionamento foi parcialmente adoptado para a investigação realizada, com a diferença que essa mensagem é utilizada para identificação do objecto. Cada emissor, envia uma mensagem própria e única que serve de identificador. Em contraste com os sistemas de localização, a investigação realizada foca-se em dar resposta a outro tipo de situações. Parte-se do princípio que a informação a transmitir se encontra previamente disponível no dispositivo de armazenamento de dados, e que os ambientes são estáticos.

Capítulo 3

Tecnologias

Este capítulo é uma análise das principais tecnologias utilizadas no desenvolvimento do projecto, para que a compreensão da solução final se torne mais simples e se percebam algumas das escolhas tomadas.

Os destaques vão para a aquisição da informação dos dados geográficos, a sua tradução para ficheiros áudio e por último o processo da transmissão ao utilizador.

3.1 GPS

O lançamento do satélite *Sputnik* por parte da Rússia, em 1957, gerou muito interesse junto da comunidade cientifica dos Estados Unidos da América. Uma equipa de cientistas norte americanos descobriu, enquanto monitorizavam as transmissões deste satélite, que os dados recebidos sofriam um desvio de frequência devido ao efeito de Doppler. Assim quando o satélite se aproximava, ou afastava a frequência aumentava, ou diminuía, respectivamente. Com estes dados era possível criar uma tabela de distorção da frequência pelo efeito de Doppler permitindo, assim, localizar o satélite na sua órbita, uma vez que a posição de referência era constante.

Mais tarde o Departamento de Defesa dos Estados Unidos da América (DoD) desenvolveu o *GPS (Global Positioning System)*, do qual possui controlo total sobre o seu funcionamento.

O GPS é um sistema de rádio navegação assente numa rede de satélites geoestacionários, que fornece todos os dados necessários para o cálculo da posição e da velocidade, no espaço e no tempo, de um utilizador, com elevado grau de precisão.

O princípio de funcionamento do GPS assenta na difusão de sinais de radio frequência, com diversos códigos e mensagens de navegação. Esta informação é processada pelo receptor permitindo calcular a posição do utilizador através do atraso na chegada das mensagens, devido à distância entre os satélites e o receptor. Como se trata de uma comunicação unidireccional e o processamento dos dados é executado pelo receptor, este sistema permite a sua utilização por um número ilimitado de utilizadores. Por outro lado, ao tratar-se de uma rede de satélites, a cobertura deste sistema é global, funcionando em qualquer meio (terra, mar, ou ar), suportando também, condições climatéricas adversas.

Trata-se de um sistema complexo, que se divide em três componentes (Figura 3.1): uma Espacial; e duas Terrestres. A componente Espacial é composta pela rede de satélites. As centrais de controlo e monitorização, e os utilizadores constituem as componentes terrestres.

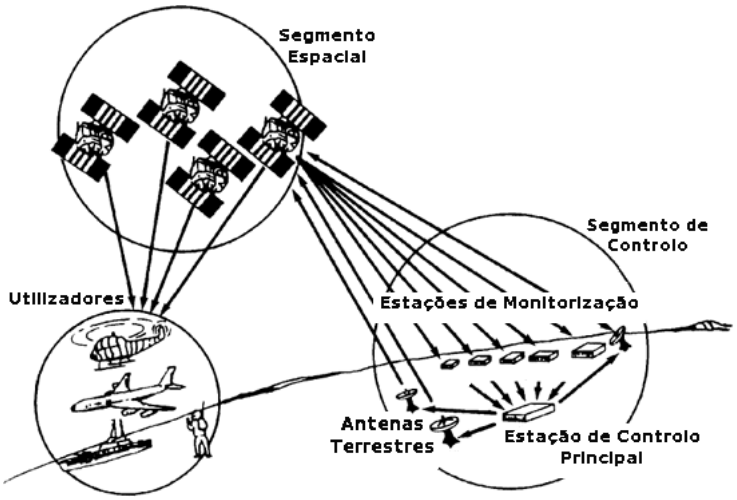


Figura 3.1: Estrutura do sistema GPS.

3.1.1 Componente Espacial

Uma constelação de 24 satélites [9] compõe a componente espacial deste sistema que apresenta uma precisão previsível de 22m no plano horizontal e de 27,7m no plano vertical no segmento de uso civil. O segmento militar - de uso restrito - apresenta melhores performances (13m e 22m nos planos horizontal e vertical respectivamente). Os satélites estão dispostos em seis planos orbitais distintos e equidistantes entre si (Figura 3.3), cada um deles inclinado 55º relativamente ao plano equatorial. São composto por quatro satélites,

que se deslocam a uma velocidade de 3,9 quilómetros por segundo, completando uma revolução em 11 horas e 59 minutos. As órbitas encontram-se aproximadamente a 26.600km do centro de massa da Terra.

Esta disposição garante que em qualquer ponto da Terra existam pelo menos 4 satélites em linha de vista, o mínimo para que se obtenha um medição precisa.

Figura 3.2: Constelação de satélites GPS.

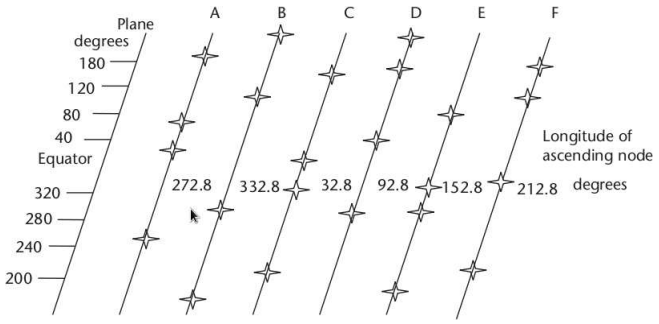


Figura 3.3: Constelação planar de satélites GPS [9].

3.1.2 Componente Terrestre

3.1.2.1 Centrais de Controlo e Monitorização

O controlo dos satélites é efectuado na base aérea norte-americana de Schiriever, onde se encontra localizada a estação responsável pelo controlo da constelação dos satélites GPS. Existem ainda outras seis estações de monitorização que recolhem, quer dados dos satélites, quer dados climatéricos locais transmitindo-os à estação de controlo para serem

analisados e permitirem identificar eventuais anomalias que possam surgir. Com estes dados, a estação de controlo, pode assumir as suas responsabilidades de actualização de mensagens de navegação, correcção de eventuais anomalias que os satélites possam sofrer (como é o caso de desvios de órbita) e monitorizam o seu estado de saúdo gerindo o carregamentos das baterias. As mensagens da estação de controlo chegam até aos satélites por meio de uma das cinco antenas que se encontram distribuídas pelo globo.

Toda esta gestão encontra-se a cargo do Departamento de Defesa Norte-Americano, proprietário do sistema [9].

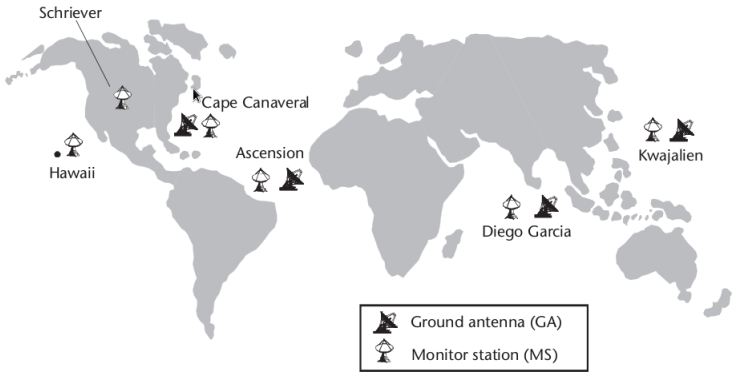


Figura 3.4: Estações de monitorização do sistema GPS [9].

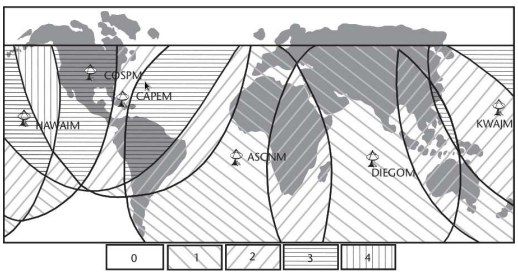


Figura 3.5: Cobertura das estações de monitorização do sistema GPS [9].

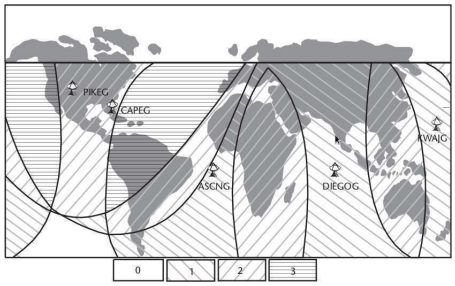


Figura 3.6: Cobertura das antenas para envio dos dados para os satélites GPS. [9].

3.1.2.2 Utilizadores

Neste segmento é constituído pela população em geral que tem acesso a um simples receptor GPS, que lhes permite a recepção dos sinais enviados pela rede de satélites. À

excepção de utilizações mais específicas, os dados depois de processados são apresentados ao utilizador sob a forma de localização geográfica, velocidade e tempo. Em termos comerciais a oferta de GPS é enorme, sendo várias as marcas que oferecem diferentes modelos com funcionalidades que procuram abranger a maior cota de mercado possível.

3.1.3 NMEA

A *NMEA (National Marine Electronics Association)* desenvolveu um conjunto de especificações que regem a comunicação com os dispositivos electrónicos de navegação. Estas especificações permitem a uniformização das comunicações com os computadores ou outro tipo de dispositivos. O protocolo - designado por *NMEA 0183* - usa uma comunicação de dados em série, no formato ASCII, que define o modo em como os dados são trocados entre os dispositivos num sistema de comunicação *half-duplex*. Uma comunicação *half-duplex* permite comunicar nas duas direcções, mas não permite que comunicação em simultâneo nas duas direcções.

Bit rate	4800
Data bits	8
Parity	*None*
Stop bits	1
Handshake	*None*

Tabela 3.1: Configuração típica da comunicação de dados.

3.2 Áudio

O áudio desde sempre se afirmou como um meio essencial na difusão de informação. Porém o armazenamento de dados deste formato digitalmente, requer elevadas capacidades de armazenamento. Os *codecs* de áudio procuram fazer frente à necessidade de comprimir estes ficheiros.

3.2.1 MP3

O *MP3* designado tecnicamente como *MPEG-1* ou *MPEG-2 Audio Layer 3*, é um formato de codificação de dados áudio que apesar de ser abrangido por algumas patentes de várias empresas, estas aceitaram licenciar serviços que usem esta especificação a preço razoáveis [10].

Trata-se de um algoritmo de compressão que abdica de informação para diminuir o tamanho ocupado pelo ficheiro. Tipicamente um ficheiro áudio de um CD de música fica em média 11 vezes mais pequeno, quando codificado a um ritmo de 128kbit/s. O principio de funcionamento deste algoritmo, é o corte das frequências dos ficheiros áudio que o ouvido humano não consegue detectar.

Vozes críticas a este formato defendem que a qualidade despendida é elevada, e se no mundo audiófilo esta discussão até pode fazer sentido, no âmbito de certas aplicações e nomeadamente da investigação realizada, esta ferramenta trata-se de uma grande mais valia pela sua eficiência no aumento da capacidade de armazenamento de informação no dispositivo. Ao ser um formato muito comum no uso diário, permite a compatibilidade da informação com outros dispositivos.

3.3 Comunicações Série

Muitas das vezes existe a necessidade dos microcontroladores trocarem informação com os periféricos. Esta informação pode ter trocada por uma de duas maneiras: ligações paralelas; ou ligações em série [11].

Pela técnica paralela, os dados, são enviados de uma forma simultânea do transmissor para o receptor. Trata-se de uma técnica muito eficiente do ponto de vista temporal, mas que, requer elevada sincronização entre os dispositivos e uma linha por cada bit da palavra.

O conceito de comunicações série define o processo de envio de dados bit a bit, sequencialmente, através de um canal de comunicação - muito recorrentemente um bus de dados de um dispositivo electrónico. Este tipo de comunicação é menos exigente quer em termos de sincronização de relógio, quer no custo do canal de comunicação (*bus*), quando comparado com uma ligação paralela. O uso diário deste tipo de comunicações é prática comum, devido à sua maior facilidade de implementação. Exemplo disso são as ligações de baixo custo como é o caso: do *I^2C, RS-232, SPI, USB*.

As ligações em série encontram-se divididas em duas classes, dependendo de haver ou não partilha do sinal de relógio [12].

Na classe de comunicações síncronas, o sinal de relógio, é partilhado pelas entidades que trocam os dados, ou seja, o sinal é comum a ambos os dispositivos (emissor e receptor). Todos os bits da palavra encontra-se sincronizados com o sinal de relógio, e são válidos durante um determinado intervalo de relógio. Dependendo da transição definida - positiva (*rising edge*) ou negativa (*falling edge*) - o bit é amostrado no instante em que ocorre essa transição. Neste tipo de transmissão de dados tanto é frequente o uso de *start* e/ou *stop bits*, bem como o uso de sinais dedicados para seleccionar os respectivos dispositivos. A Figura 3.7 ilustra a transferência de dados, relativamente ao sinal de relógio.

O uso de comunicações síncronas deve ter em consideração a possibilidade de existência de perturbações no sinal de relógio. Estas perturbações podem alterar o instante de amostragem, conduzindo a erros de decisão. No caso de comunicações de longa distância, os sinais estão sujeitos a três perturbações que podem levar ao efeito anterior: a atenuação; o ruído; e o atraso.

- Atenuação - é necessário garantir que o sinal de relógio chegue, com amplitude suficiente ao destino, para que o receptor consiga discriminar a alteração de nível lógico responsável por desencadear a amostragem do bit;

- Ruído - pode desfigurar de o sinal de relógio se possuir amplitude suficiente para alterar o nível lógico do relógio.

- Atraso - caso se verifique uma diferença entre os percursos dos sinais de relógio e dados significativa, podem ser amostrados bits amostrados podem ser atrasados ou adiantados.

Classificam-se como assíncronas as ligações onde não existe partilha do sinal de relógio entre o receptor e o emissor. Cada dispositivo é responsável por gerar o seu sinal de relógio. Uma transmissão de dados assíncrona, necessita de um *start bit*, para indicar o início de uma transmissão, e um ou mais *stop bits*, responsáveis por indicarem o fim da transmissão da palavra.

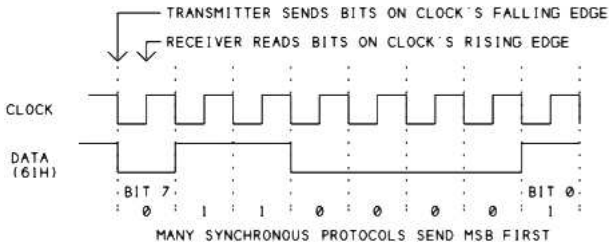


Figura 3.7: Comunicação síncrona de dados [12].

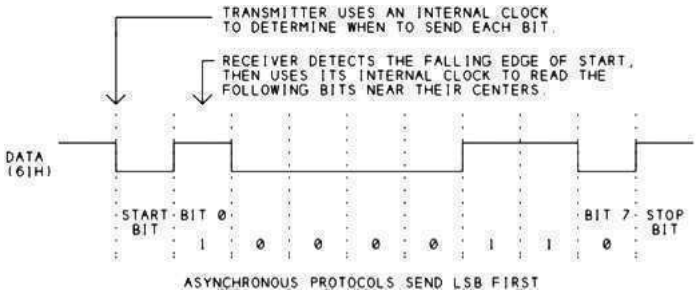


Figura 3.8: Comunicação assíncrona de dados [12].

3.3.1 SPI

O SPI (Serial Peripheral Interface Bus) foi inicialmente desenvolvido pela Motorola [13], com o intuito de fornecer uma solução de baixo custo e simples de comunicação entre microcontroladores e outros circuitos periféricos.

São muitas as empresas produtoras, quer de dispositivos periféricos, quer de microcontroladores, que adoptaram este tipo de comunicação. Isto, provoca com que a já elevada lista de dispositivos (que a suportam) continue a aumentar.

O SPI trata-se de uma ligação de dados síncrona, em que o sinal de relógio é gerado por um dispositivo principal - designado por *master* (microcontrolador). O sinal de relógio é partilhado com o dispositivo periférico - designado por *slave* - que o utiliza para sincronizar a trama de bits em série.

Numa comunicação SPI, entre um *master* e um *slave*, são necessário quatro sinais: *MOSI (Master Out Slave In)*; *MISO (Master In Slave Out)*; *SCK (Serial Clock)*; e *CS (Chip Select)*. Por cada dispositivo adicional, será preciso mais um sinal de *Chip Select* para fazer a respectiva selecção desse componente.

Na Figura 3.9 estão ilustradas as ligações *master* ↔ *slave* necessárias para uma comunicação com três periféricos, enquanto que, na Figura 3.10 estão ilustrados os sinais durante a comunicação com um desses dispositivos.

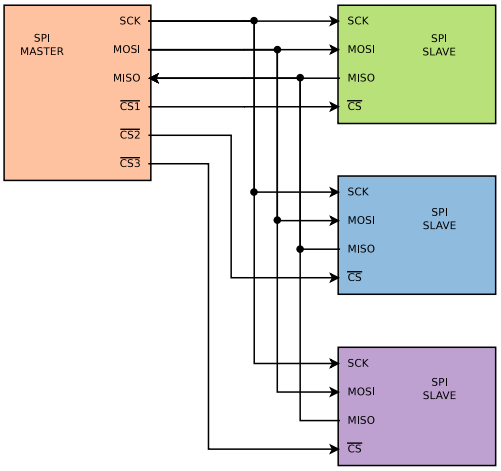


Figura 3.9: Exemplo de Comunicação SPI.

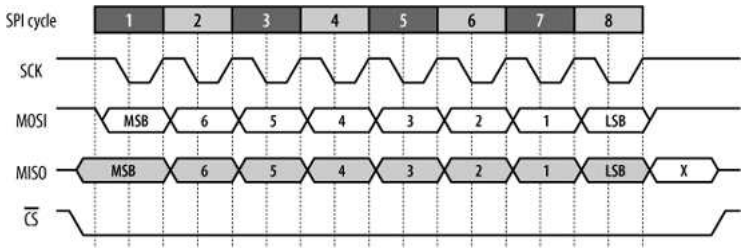


Figura 3.10: Sinais durante um ciclo de transferência de dados SPI.

3.3.2 USART

Enquanto que uma ligação de dados *Universal Asyncronous Receiver Transmitter* (UART) está limitada a um uso assíncrono, uma ligação *Universal Syncronous Asyncronous Receiver Transmitter* (USART) permite optar pela uso síncrono ou assíncrono na transferência de dados.

Na sua configuração mais típica as tramas de dados série anteriores são constituídas por 8 bits de dados e 1 stop bit, e não têm paridade. Exemplo disso é a Figura 3.11

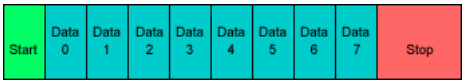


Figura 3.11: Configuração típica de uma trama de dados USART.

3.3.3 USB

O *Universal Serial Bus* (USB) é uma especificação para o estabelecimento de comunicação entre equipamentos e um controlador anfitrião (*host controller* (Figura 3.12), desenvolvido originalmente por sete companhias: Compaq, DEC, IBM, Intel, Microsoft, Nec e Nortel [14].

A principal característica desta ligação é a simples conexão e configuração dos periféricos a quando da sua ligação ao controlador anfitrião (*plug-and-play*). Isto fez com se tornasse numa ligação padrão, substituindo o uso de outras ligações série e paralelas em dispositivos como teclados, ratos, impressoras, unidades de armazenamento externo, entre outras.

A taxa de transferência de dados desta norma tem evoluído favoravelmente desde a primeira revisão (USB 1.0), em que se anunciava como taxa máxima 12Mbit/s (com largura de banda máxima - *full-bandwidth*). Na revisão, 2.0 (a mais utilizada à data) essa taxa aumentou para os 480Mbit/s. A revisão 3.0 por sua vez atinge taxas na ordem dos 4.8Gbit/s.

A todos estes factores, há que juntar ainda o facto de existir compatibilidade entre dispositivos de revisões diferentes, o que torna possivel que equipamentos de revisões diferentes comuniquem entre si.

Por todas estas razões adoptou-se esta ligação para o envio de dados para o dispositivo e programação do mesmo.

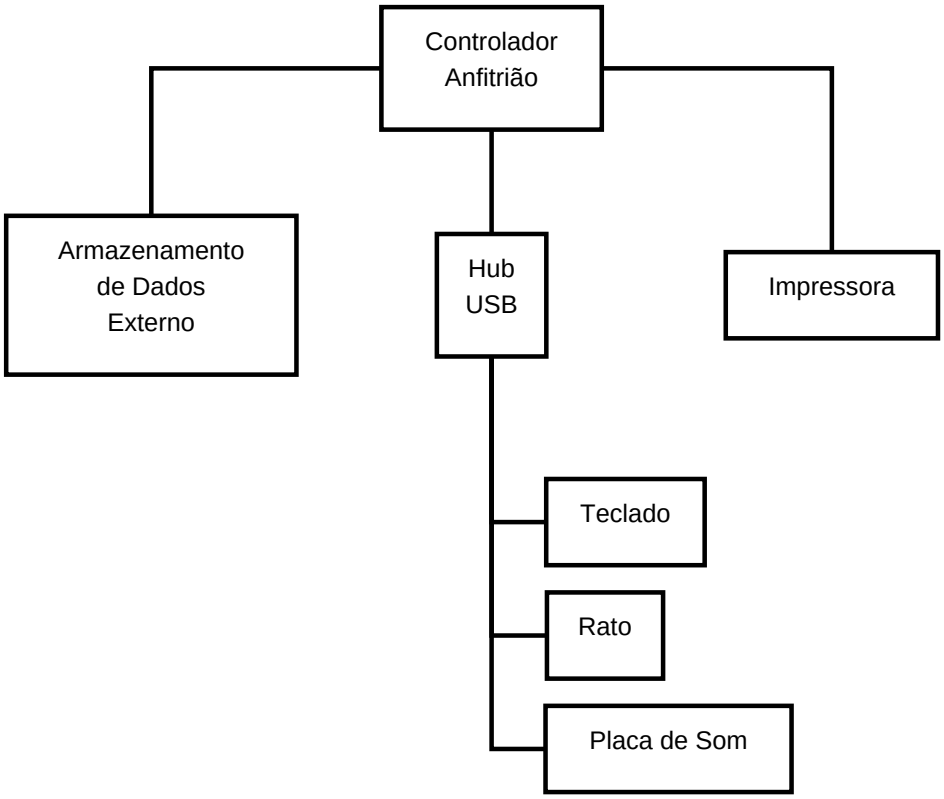


Figura 3.12: Exemplo de uma estrutura de comunicação USB.

Capítulo 4

Hardware

Neste capítulo são apresentadas as principais plataformas que serviram de suporte para o trabalho desenvolvido nesta dissertação. É dado especial relevo às principais características dos blocos constituintes do circuito final.

Inicialmente, serão apresentados detalhadamente os principais blocos, constituintes do circuito final. Posteriormente será apresentada a arquitectura do projecto, dando especial atenção à interligação dos diversos blocos.

4.1 Microcontrolador

A complexidade do projecto, devido à quantidade de entidades nele presente, estabeleceu a necessidade de existir um controlador para o fluxo de informação entre os diferentes blocos. Apesar de outros componentes utilizados possuírem microcontroladores, passiveis de executarem as funções deste bloco, optou-se por não o fazer devido a consumos de energia, e ao facto de se pretender manter uma camada de abstracção entre as tecnologias de localização e os outros componentes do sistema. Isto permite que o sistema funcione com outras tecnologias sem que para isso seja necessário alterar o código do microcontrolador, bastando apenas que estas respeitem os protocolos de comunicação: USART e NMEA-0183.

Desde logo, surgiram algumas importantes limitações na escolha final:

- Baixo consumos - trata-se de um dispositivo móvel onde a autonomia é valorizada;

- Diversos canais de comunicação (SPI, USART, USB) - para comunicação com os periféricos;
- Uma arquitectura e funcionamento simples e acessíveis ao utilizador;
- Ferramentas de desenvolvimento de baixo custo;
- Baixo custo.

A escolha recaiu pela plataforma AT32UC3B0256 [15, 16, 17, 18] da Atmel, por apresentar todas as características anteriores.

Trata-se um microcontrolador de 32bits com 256K Bytes de memória interna de alta velocidade com suporte USB 2.0, que permite o uso de aplicações livres para o desenvolvimento e programação de código.

Na Figura 4.1 podemos verificar as principais características deste microcontrolador.

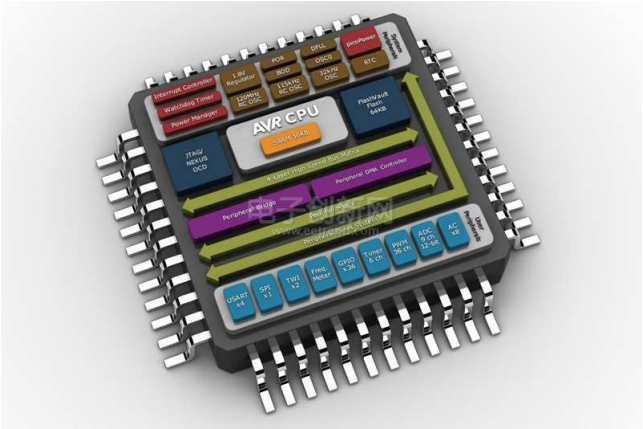


Figura 4.1: Representação da arquitectura do microcontrolador

4.2 Módulo GPS ORG-1318

O módulo ORG-1318 4.2 da Origin [19] é um receptor GPS multi-canal, com antena microstrip incorporada, construído sobre um chipset SiRFstarIII [20]. Este módulo é compatível com dois protocolos de comunicação com outros dispositivos: o NMEA-0183; e o SiRF binary protocol. Possui a flexibilidade de permitir a comunicação através de uma

ligação USART, ou de uma ligação SPI. As principais funcionalidades deste componente são descritas na Tabela 4.1.

Optou-se por utilizar uma configuração de comunicação UART, para libertar um pouco o BUS SPI, para outros dispositivos.

Solução integrada GPS multi-canal
Antena Patch Microstrip integrada
Chip SiRFstarIII GSC3LTf
20 canais em modo de pesquisa
12 canais em paralelo para modo de localização
Sensibilidade mínima para aquisição de -157dBm
Sensibilidade mínima no modo de localização -159dBm
Tempo de arranque típico inferior a 35s
Modos Automáticos e programáveis de poupança de energia
Baixo consumo de energia: 100mW durante a aquisição do sinal
Interfaces de comunicação: UART ou SPI
Protocolo UART e taxa de transferência programáveis
Protocolos de comunicação standart: NMEA-0183 ou SiRF Binary
Gama de tensão de alimentação: 3.3V a 5.5V
Dimensões reduzidas: 17mm x 17mm
Dispositivo de montagem na superfície (SMD)
Gama de temperatura de funcionamento: -400 to 850ºC
Pb-Free RoHS compliant

Tabela 4.1: Principais características do SiRFstarIII GSC3LTf [21].

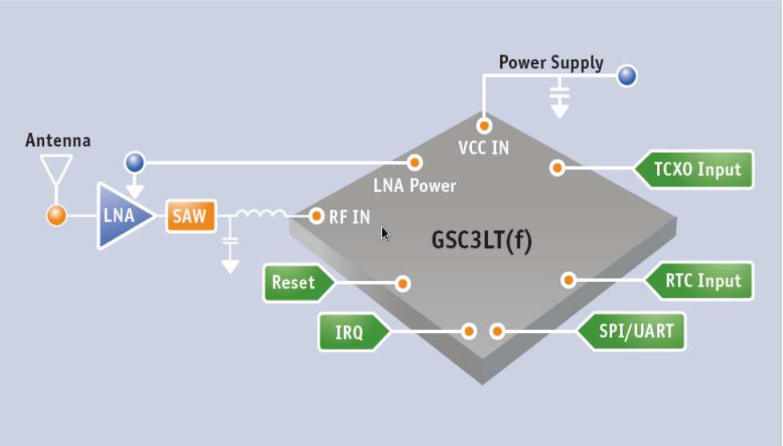


Figura 4.2: Arquitectura do módulo GPS [21].

4.3 Sensor ZigBee

O CC2430/31 é um módulo especialmente criada para redes sem fios de baixo consumo baseadas em IEEE 802.15.4 e ZigBee. Este módulo contém um transceptor CC2420, um micro-controlador de baixo consumo 8051, 8kB de RAM, 128kB de memória flash e muitas outras características.

Figura 4.3: Módulo ZigBee utilizado

4.4 Transmissor FM

O transmissor FM é um módulo de reduzidas dimensões 4.4, desenhado para ser incorporado em microfones wireless e transmissores de MP3 de baixo custo. Necessita apenas de dois componentes externos para o seu funcionamento, e apresenta um interface completamente digital de controlo.

A Tabela 4.2 apresenta as características deste módulo.

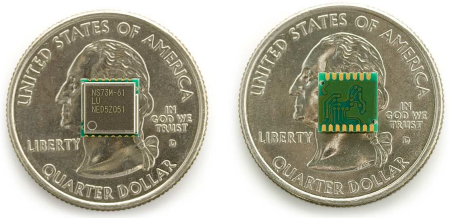


Figura 4.4: Módulo transmissor FM.

Gama de frequência dos 87.5MHz aos 108MHz
Modelação multiplexada Stereo
Oscilador externo: 32.768kHz hline Reduzidas dimensões: 7x7x1.5mm
Suporte para interface SPI ou I2C
Gama de tensões de alimentação: 2.7V to 3.6V
Gama de temperaturas suportadas: -20C to 75C operation

Tabela 4.2: Principais características do transmissor FM [22, 23, 24].

4.5 Descodificador Áudio

A escolha recaiu pelo VS1053 da VLSI, que entre outros formatos, descodifica MP3. Trata-se da solução mais versátil da VLSI, que para além de ser capaz de descodificar vários codecs, é também de capaz de gravar ficheiros áudio em três formatos diferentes. Em resumo, é um poderoso codificador/descodificador áudio de fácil utilização.

Lista das principais funcionalidades:

- Descodificação de vários formatos:
  - Ogg Vorbis;
  - MP3 - MPEG 1 & 2 audio layer III (CBR+VBR+ABR);
  - MP1 & MP2 - MPEG 1 & 2 audio layers I & II opcionais;

  - MPEG4/2 AAC-LC(+PNS), HE-AAC v2 (Level 3) (SBR + PS);
  - WMA4.0/4.1/7/8/9 (5-384 kbps);
  - FLAC áudio sem percas através de software adicional (até 24 bits, 48 kHz);
  - WAV (PCM + IMA ADPCM);
- Codifica três formatos diferentes a partir da entrada de linha ou do microfone:
  - Ogg Vorbis com software adicional;
  - IMA ADPCM;
  - 16-bit PCM;
- Suporte para streaming;
- Controlo de agudos e de baixos;
- Necessidade de apenas um sinal de relógio externo 12-13 MHz ou 24-26 MHz clock;
- Operação em baixo consumo;
- Capacidade de drive de auscultadores estereofónicos;
- Acesso a 16.5 KiB on-chip RAM para código e dados;
- Controlo por porta série e interface de dados;
- UART para processo de debug;
- Possibilidade de adicionar novas funcionalidade por software e 8 pinos gerais de IO.

Capítulo 5

Software e Programação

Neste capítulo são apresentadas as ferramentas de software usadas no desenvolvimento do projecto, desde o desenvolvimento do protótipo à sua programação.

5.1 Desenho da Placa de Circuito Impresso

O processo de desenho e fabrico de placas de circuito impresso tem sofrido constantes evoluções ao longos dos anos.

Com a evolução tecnológica que se tem verificado, surge a necessidade de ter processos cada vez mais eficazes para o fabrico de placas de circuito impresso, que implicam sobretudo uma maior definição na construção das mesmas. Esta definição, na prática, traduz-se na diminuição das distâncias mínimas que é possível obter com uma determinada tecnologia.

Face à necessidade anterior, surge paralelamente uma outra: a procura por ferramentas que tornem todo este processo mais simples e eficaz para o utilizador.

É aqui que surgem as aplicações CAD (*Computer-Aided Design*). Trata-se de aplicações nas quais se esquematizam e desenham os circuitos com o auxílio do computador.

Actualmente a oferta de aplicações para este fim é muito grande, pois neste tipo de aplicações, em que o interface com o utilizador assume um papel crucial na sua escolha, a procura é muito diversificada. Para além disso, há que ter também, em consideração as funcionalidades que se pretendem, e que podem justificar a opção por um programa mais complexo.

5.1.1 Altium Designer

O Altium Designer [25], é uma ferramenta de ambiente gráfico Windows, para desenho e simulação de circuitos electrónicos. Tem uma interface apelativa, eficiente e simples para esse fim.

Por defeito, vem acompanhado com bibliotecas dos fabricantes de maior renome (com mais de 100.000 componentes), o que simplifica bastante o processo. Independentemente disso, o utilizador pode ainda desenhar os seus próprios componentes.

Uma utilização eficaz desta ferramenta (e de outras), impõe que o utilizador, desenhe o esquema eléctrico primeiro, e construa a placa em consequência deste. O esquema é mais fácil de verificar, e transmite maior segurança ao utilizador para evitar erros no processo seguinte. Assim, a placa reflecte o que existe no esquema, permitindo evitar erros como a troca de pinos, o estabelecimento de ligações erradas, ou a existência de curto-circuitos.

A placa deverá assim verificar um conjunto de regras (*DRC - Design Rule Check*) que impedem a ocorrência de alguns erros. Para além das regras por defeito, como são as definições do fabricante, o Altium permite ainda criar novas regras com a ajuda de uma representação gráfica, que torna o processo muito mais intuitivo para o utilizador.

De entre muitas outras funcionalidades, há que destacar as seguintes:

- Simulação 3D das placas - permite ter uma ideia do aspecto final da placa;
- Automatização dos processos de implantação dos componentes e das pistas;
- Automatização dos processos de enumeração dos componentes - que permitem ao utilizador abstrair-se deste processo e impedindo que não hajam referências repetidas, ou seja, componentes com a mesma referência - muito útil em placas com elevado número de componentes e/ou páginas com o esquema;
- Histórico de modificações - permite ao utilizador manter um registo das alterações que vai efectuando;
- Controlo de versões de placas - permite produzir placas diferentes do mesmo esquema sem que seja necessário um novo projecto;
- Suporte para vários formatos tipo suportados pelos fabricantes (incluído o formato Gerber RS-274X).

O Altium Designer revelou-se uma ferramenta muito útil, com uma rápida curva de aprendizagem, e apesar de se tratar de uma versão experimental, dá acesso a todas as suas funcionalidades, bem como aos guias em vídeo de treino.

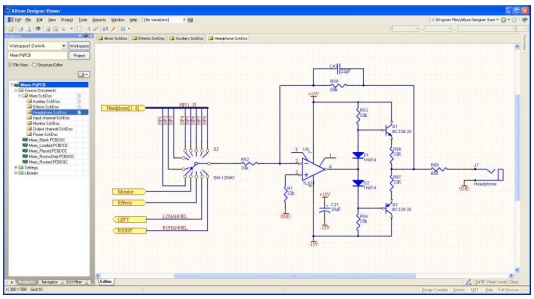


Figura 5.1: Altium no modo de edição do esquema.

Figura 5.2: Altium no modo de edição do PCB.

5.1.2 Formato Gerber RS-274X

Com o elevado número de aplicações existentes, e porque na sua generalidade cada uma tem um formato próprio de ficheiros, surgiu a necessidade de criar alguns formatos tipo para que os fabricante pudessem suportar o maior número de clientes possível.

A Eurocircuits [26], no seu guia de regras [27], dá preferência ao formato RS-274X [28], pelo que foi este o formato escolhido para o envio do desenho das placas para o fabricante.

O formato Gerber é um conjunto caracteres ASCII, com a informação das coordenadas (segundo os eixos X e Y) e dos comandos a executar [29].

É ainda necessário o envio de um ficheiro com a informação da furação da placa, que para além das coordenadas dos furos, identifica também as ferramentas necessárias para efectuar essa mesma furação.

5.2 Desenvolvimento da Aplicação

5.2.1 C

A linguagem de programação *C*, surgiu no inicio dos anos 70, desenvolvida por Ken Thompson, Dennis Ritchie, entre outros investigadores dos laboratórios Bell [30].

Quando Thompson desenvolveu o sistema operativo Unix original (totalmente escrito

em linguagem de máquina - *assembly*), surgiu a necessidade de uma nova linguagem de programação que facilitasse o processo de debug e melhoramento do código, até então processos muito complicados. É então que adoptam uma nova linguagem de programação - B. No entanto, com a aquisição de um PDP-11 para correr o sistema, apercebem-se que esta linguagem não conseguia satisfazer as suas necessidades, optando por a desenvolverem e aperfeiçoar às medidas das suas necessidades. Em 1973, surge assim, o C como uma linguagem madura e estável o suficiente para ser utilizada.

Actualmente, continua a ser amplamente usada, especialmente para o desenvolvimento de sistemas com microcontroladores, por se tratar de uma linguagem simples e intuitiva, e apresentar uma boa relação desempenho vs eficiência.

Na generalidade, a grande maioria dos fabricantes fornecem ferramentas para o desenvolvimento de aplicações nesta linguagem de programação, sendo quase um standart, a sua utilização.

O *C* serviu também como ponto de partida para outras linguagem que começam a aparecer para o desenvolvimento de aplicações como é o caso do *C++* e do *C#*.

5.2.2 AVR32 Studio

O AVR32 Studio (Figura 5.3), é um ambiente de desenvolvimento gráfico multi-plataforma (IDE), fornecido pela Atmel. Baseado no Eclipse [31], é também uma aplicação gratuita e de código aberto que permite o desenvolvimento das aplicações em C e/ou C++.

Inclui os compiladores de C e C++ - baseados no GCC [32] - necessários para compilarem o código, também eles grátis e com o código aberto.

Esta poderosa ferramenta, permite fazer depuração do código quer localmente, quer no circuito, através de um dos vários dispositivos suportados. Pode-se ainda efectuar directamente a programação do circuito através do *bootloader*, sendo para isso necessário o *dfu-programmer*.

5.2.2.1 AVR UC3 Software Framework

Na última versão do AVR32 Studio (2.6) vem incluída, por defeito, a AVR UC3 Software Framework [34]. Trata-se de um conjunto de bibliotecas, escritas em C, muito bem documentadas, que permitem ao utilizador controlar e aceder aos periféricos, do microcontrolador da Atmel com um núcleo UC3.

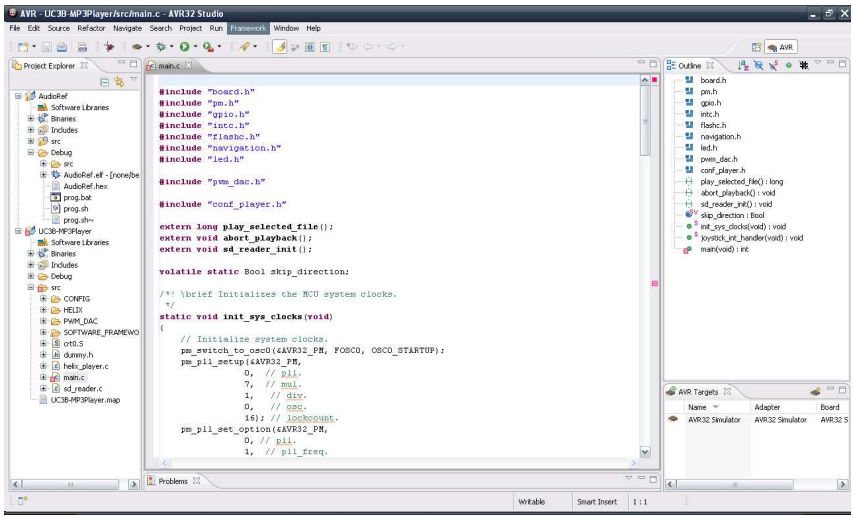


Figura 5.3: Programa de desenvolvimento AVR32 Studio [33].

Este interface permite ao utilizador abstrair-se da configuração e inicialização dos serviços, e periféricos do microcontrolador, em baixo nível.

Para além disso, inclui também um vasto conjunto de projectos exemplo que fazem uso dos diversos módulos do microcontrolador, para o utilizador se ambientar com os mesmos.

5.3 Programação do Microcontrolador

A programação dos microcontroladores pode ser feita de variadas maneiras, nomeadamente por ISP fazendo uso do *bootloader* pré-programado no microcontrolador ou por JTAG (o que requer um dispositivo adicional compatível com esta norma e suportado pelo AVR32 Studio).

O *bootloader* é um pequeno bloco de código, referenciado no inicio da memória (Figura 5.4), que decide o que fazer quando o microcontrolador arranca.

Caso lhe seja dada indicação (numa janela temporal definida) de que se pretende programar o dispositivo, ele fica a aguardar pelo envio do ficheiro, caso contrário arranca com o programa que possui em memória. A indicação pode ser dada de várias formas: quer pelo envio de um código pelo porto de comunicação de dados com o *bootloader*; quer por sinalização num pino de I/O do dispositivo. Todos estes parâmetros são definidos pelo utilizador. Para se tirar partido do *bootloader* o dispositivo tem que suportar auto-programação

da memória.

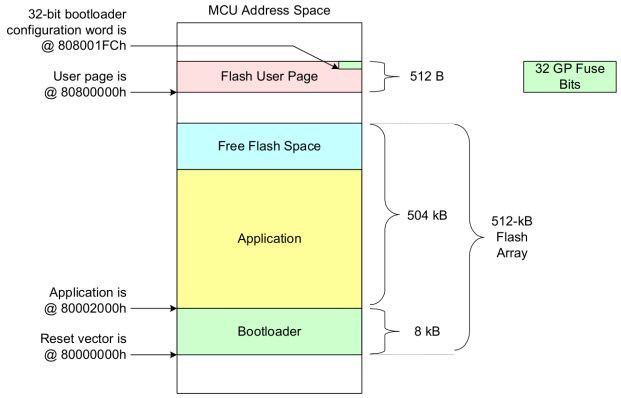


Figura 5.4: Organização da memória do microcontrolador com o bootloader.

5.3.1 Utilização do dfu-programmer

O *dfu-programmer* [35] trata-se de uma pequena aplicação de linha de comandos para programar microcontroladores da Atmel, com suporte para programação ISP via porto USB.

É uma aplicação livre e muito simples, desenhada para sistemas Unix, e que que tira partido do *bootloader* para o envio dos dados.

A Tabela 6.3 apresenta os comandos necessários para a correcta programação de um dispositivo (Atmel AT32UC3B0256). O primeiro comando apaga o conteúdo desse componente, caso exista algum. O segundo carrega para a memória flash o ficheiro indicado. Por último o terceiro força o arranque do microcontrolador.

Devido à possibilidade de existência de lixo na memória *flash*, é aconselhável proceder sempre à limpeza desta, sobre pena de o funcionamento do programa carregado ser comprometido.

```
1 dfu-programmer at32uc3b0256 erase
  dfu-programmer at32uc3b0256 flash --suppress-bootloader-mem ficheiro.hex
3 dfu-programmer at32uc3b0256 reset
```

Tabela 5.1: Comandos para a programação o microcontrolador através do dfu-programmer.

Capítulo 6

Trabalho Desenvolvido

6.1 Decisões Arquitectónicas do Circuito

O protótipo final, devido à sua complexidade e ao elevado número de componentes, foi desenhado de forma a apresentar uma dimensão o mais reduzida possível. Foi por isso, necessário recorrer á colocação de componentes em ambas as faces da placa. Pelas mesmas razões, deu-se preferência ao uso de componentes SMD, sempre que possível, por ocuparem menor área e não afectarem necessariamente a camada contrária.

Desde inicio, por existirem componentes de rádio-frequência envolvidos, idealizou-se a sua colocação na parte superior da placa, para prevenir ao máximo interferências inerentes à sua posição. Como esta zona se trata também da zona passível de se encontrar mais descoberta, foi aqui que se decidiu colocar os leds para transmitir a algumas informações importantes ao utilizador, como o estado dos módulos (ZigBee, GPS, transmissor FM, e estado de funcionamento).

Em todos as áreas procurou-se satisfazer as especificações dos fabricantes para garantir correcto funcionamento do equipamento.

O projecto CAD do protótipo final pode ser analisada nas Figuras 6.1 e 6.2. Enquanto que a Figura 6.3 é uma fotografia do protótipo montado.

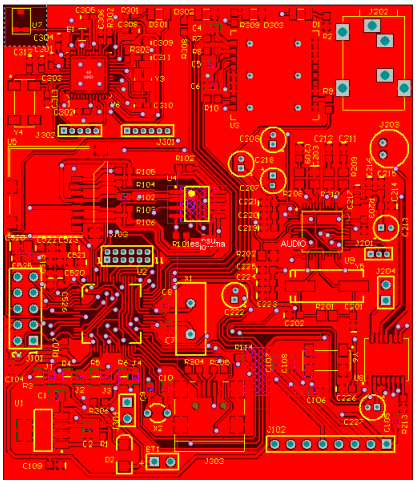


Figura 6.1: Face superior do projecto CAD.

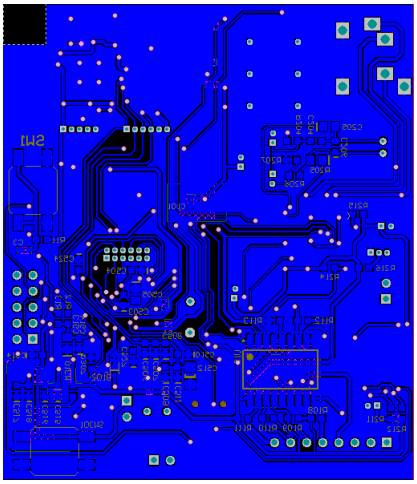


Figura 6.2: Face inferior do projecto CAD.

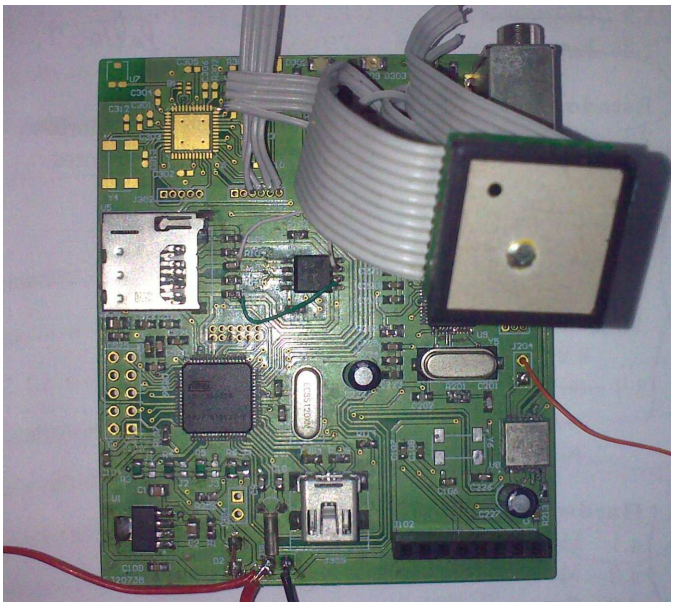


Figura 6.3: Organização da memória do microcontrolador com o bootloader.

6.1.1 Estrutura do protótipo

Analisando a solução final obtida, verifica-se que os módulos de GPS e ZigBee assumem o papel de recepção da informação geográfica. Esses dados são recebidos e alimentados à

unidade de controlo central através de dois canais de comunicação *USART* independentes.

O módulo ZigBee tem ao seu dispor um led, do qual tem absoluto controlo para indicar o seu estado de funcionamento. É também responsável pela sinalização junto do microcontrolador da existência de informação a ser processada.

O led indicador do estado do GPS é comandado pelo microcontrolador central, uma vez que é esta identidade que assume a gestão de energia deste módulo. O microcontrolador tem ainda a seu cargo a responsabilidade pela verificação da existência de informação que justifique ser analisada.

A comunicação entre com os outros componentes (teclado, cartão de memória e circuito de áudio) é feita pela ligação de dados SPI e gerida pelo microcontrolador central.

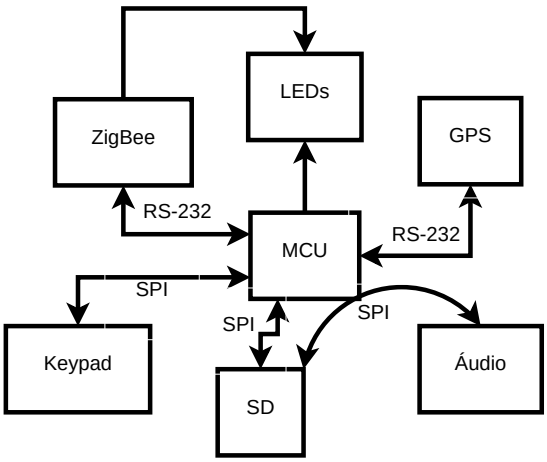


Figura 6.4: Bloco organizacional do sistema.

6.2 Solução Final

Como já referido, existem duas fontes de informação: GPS e ZigBee.

Quando o utilizador se encontra no exterior, o GPS é a solução adoptada, que quando activado, tenta automaticamente receber informação GPS. O módulo começa, desde logo, a enviar informação para o microcontrolador pelo porto USART com os valores do protocolo NMEA-0183. Até conseguir estabelecer ligação à rede, a informação sai do módulo a zeros. O microcontrolador ao receber esta informação descarta os dados uma vez que os mesmos não fazem sentido.

Quando os dados passam a ser coordenadas válidas o microcontrolador usa os esses dados para procurar referências na redondeza. O ficheiro a usar é seleccionado da pasta com o idioma definido.

O código desenvolvido encontra-se em anexo, mas alguns blocos vão ser analisados devido à sua importância na compreensão do funcionamento do sistema. As funções de interacção com os periféricos do microcontrolador, por fazerem parte da plataforma de desenvolvida, podem ser consultadas na documentação da mesma [34].

Em anexo encontra-se também o ficheiro onde se encontram definidos os serviços e definições do circuito

A Figura 6.5 é um esquema do fluxo de código que o microcontrolador executa. Ao longo das próximas secções este vai ser analisado, dando especial atenção ao código responsável pela execução de cada componente.

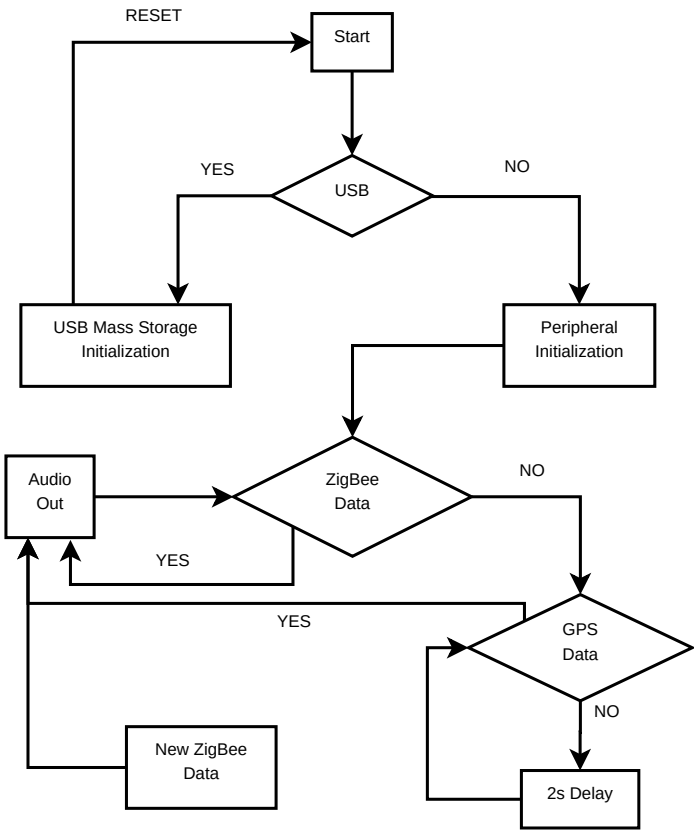


Figura 6.5: Bloco organizacional do sistema.

6.2.1 Inicialização dos dispositivos

Os primeiros passos do microcontrolador é inicializar alguns periféricos indispensáveis como é o caso dos sinais de relógio.

```
int _init_startup(void)
{
  // Import the Exception Vector Base Address.
  extern void _evba;

  // Load the Exception Vector Base Address in the corresponding system
     register.
  Set_system_register(AVR32_EVBA, (int)&_evba);

  // Enable exceptions.
  Enable_global_exception();

  // Initialize interrupt handling.
  INTC_init_interrupts();

  // Give the used CPU clock frequency to Newlib, so it can work properly
     .
  set_cpu_hz(pcl_freq_param.pba_f);

  // Don't-care value for GCC.
  return 1;
}
```

Tabela 6.1: Inicialização dos serviços mínimos microcontrolador.

A fase seguinte contempla a verificação da existência ou não de ligação de dados USB. No caso de tal se verificar o microcontrolador dá início ao processo de configuração dos periféricos necessários para uma comunicação de dados com o computador. Esta situação verifica-se por exemplo quando se pretende enviar novos dados para o cartão de memória.

```
#if USB_MODE_START == ENABLED
          // Initialize USB clock.
          pcl_configure_usb_clock();
          // Initialize USB tasks.
          usb_task_init();
        #if USB_DEVICE_FEATURE == ENABLED
          device_mass_storage_task_init();
        #endif

        ushell_task_init(pcl_freq_param.pba_f);

        #ifdef FREERTOS_USED
          vTaskStartScheduler();
          portDBG_TRACE("FreeRTOS returned.");
          return 42;
        #else
          while (TRUE)
          {
            usb_task();
          #if USB_DEVICE_FEATURE == ENABLED
            device_mass_storage_task();
          #endif
          #if USB_HOST_FEATURE == ENABLED
            host_mass_storage_task();
          #endif
            ushell_task();
          }
        #endif  // FREERTOS_USED
        #else
          while(1);
        #endif
```

Tabela 6.2: Inicialização dos serviços USB.

```
/*! \brief Initializes SD/MMC resources: GPIO, MCI and SD/MMC.
 */
static void sd_mmc_mci_resources_init(void)
{
  static const gpio_map_t SD_MMC_MCI_GPIO_MAP = {
    {SD_SLOT_8BITS_CLK_PIN,   SD_SLOT_8BITS_CLK_FUNCTION  },
    {SD_SLOT_8BITS_CMD_PIN,   SD_SLOT_8BITS_CMD_FUNCTION  },
    {SD_SLOT_8BITS_DATA0_PIN, SD_SLOT_8BITS_DATA0_FUNCTION},
    {SD_SLOT_8BITS_DATA1_PIN, SD_SLOT_8BITS_DATA1_FUNCTION},
    {SD_SLOT_8BITS_DATA2_PIN, SD_SLOT_8BITS_DATA2_FUNCTION},
    {SD_SLOT_8BITS_DATA3_PIN, SD_SLOT_8BITS_DATA3_FUNCTION},
    {SD_SLOT_8BITS_DATA4_PIN, SD_SLOT_8BITS_DATA4_FUNCTION},
    {SD_SLOT_8BITS_DATA5_PIN, SD_SLOT_8BITS_DATA5_FUNCTION},
    {SD_SLOT_8BITS_DATA6_PIN, SD_SLOT_8BITS_DATA6_FUNCTION},
    {SD_SLOT_8BITS_DATA7_PIN, SD_SLOT_8BITS_DATA7_FUNCTION}
  };

  // MCI options.
  static const mci_options_t MCI_OPTIONS = {
    .card_speed = 400000,
    .card_slot  = SD_SLOT_8BITS, // Default card initialization.
  };

  // Assign I/Os to MCI.
  gpio_enable_module(SD_MMC_MCI_GPIO_MAP, sizeof(SD_MMC_MCI_GPIO_MAP) /
      sizeof(SD_MMC_MCI_GPIO_MAP[0]));

  // Enable pull-up for Card Detect.
  gpio_enable_pin_pull_up(SD_SLOT_8BITS_CARD_DETECT);

  // Enable pull-up for Write Protect.
  gpio_enable_pin_pull_up(SD_SLOT_8BITS_WRITE_PROTECT);

  // Initialize SD/MMC with MCI PB clock.
  sd_mmc_mci_init(&MCI_OPTIONS, pcl_freq_param.pba_f, pcl_freq_param.
      cpu_f);
}
```

Tabela 6.3: Inicialização da memória flash.

6.2.2 Aquisição de Dados Geográficos

Os dados adquiridos pelas duas tecnologias de localização, são periodicamente verificados pelo microcontrolador do sistema. Devido à diferença de funcionamento dos dois protocolos existem algumas diferenças na forma em como estas entidades são analisadas.

6.2.2.1 GPS

O módulo GPS envia a informação periodicamente pelo porto USART usando o protocolo NMEA. Esta informação, quer pela margem de erro do sistema GPS, quer pelo facto das mensagens de áudio terem uma duração grande quando comparada com o período de transmissão de dados entre o GPS e o microcontrolador, não necessita de ser analisada na sua totalidade. Assim, o microcontrolador, só consulta os dados em duas situações:

- Após a reprodução de uma mensagem de áudio, pois é possível que existam novos dados a serem reproduzidos;

- Ou em ciclos de 2 segundos gerados pelo timmer X.

Com esta solução o microcontrolador não só, fica mais tempo livre para executar outras funções, como se diminui o consumo de energia, uma vez que este passa mais tempo sem actividade.

6.2.2.2 ZigBee

O funcionamento do módulo ZigBee, é algo diferente do módulo GPS, uma vez que este possuí um microcontrolador que é facilmente acedido para programação. O módulo procura por referências de emissores. Quando surge nova informação, esta é transmitida pelo porto série, gerando uma interrupção junto do microcontrolador, que recebe esses dados e os processa.

O código do ZigBee que permite executar este processo encontra-se descrito na Tabela 6.4.

```
// Waiting for a message
while ((MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive(RefNode_TaskID
    )))
{
        // check event
        switch ( MSGpkt->hdr.event ) {
                // Message Received
                case AF_INCOMING_MSG_CMD:
                        // Send message to USART
                        usart_send(pkt->srcAddr.addr.shortAddr);
        }
}
```

Tabela 6.4: Código do Módulo ZigBee.

6.2.3 Processamento dos Dados Adquiridos

As diferenças, entre os dados adquiridos, traduzem-se também no sistema de reprodução dos ficheiros sonoros. Como a origem dos dados é diferente, e os dados em si também (coordenadas no caso do GPS e identificadores no caso do ZigBee), são necessários dois blocos para processar essa informação.

Existem dois ficheiros de dados, que são índices que ligam os dados aos ficheiros áudio disponíveis no cartão de memória. Na Tabela 6.5, está ilustrado uma entrada do ficheiro das referências do GPS, enquanto que a Tabela 6.6 é o mesmo exemplo para o caso de uma referência do módulo ZigBee.

```
<!-- XML Header -->
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- POI example -->
<gpx xmlns="http://www.topografix.com/GPX/1/1">
<wpt lat="40.62946331750202" lon="-8.656012693439623">
<name>IT</name>
<desc/>
<link href="Data/Location_0"/>
<sym>Waypoint</sym>
<extensions>
<gpxx:WaypointExtension xmlns:gpxx="http://www.garmin.com/xmlschemas/
    GpxExtensions/v3">
<gpxx:Proximity>20</gpxx:Proximity>
<gpxx:DisplayMode>SymbolAndName</gpxx:DisplayMode>
</gpxx:WaypointExtension>
</extensions>
</wpt>
<!-- New entries here! -->
<!-- The End -->
</gpx>
```

Tabela 6.5: Código XML de um ficheiro com dados do GPS.

```
<!-- XML Header -->
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- POI example -->
<gpx xmlns="http://www.topografix.com/GPX/1/1">
<wpt id="1234566778">
<name>IT</name>
<desc/>
<link href="Data/Location_0"/>
<sym>Waypoint</sym>
<extensions>
<gpxx:WaypointExtension xmlns:gpxx="http://www.garmin.com/xmlschemas/
    GpxExtensions/v3">
<gpxx:DisplayMode>SymbolAndName</gpxx:DisplayMode>
</gpxx:WaypointExtension>
</extensions>
</wpt>
<!-- New entries here! -->
<!-- The End -->
</gpx>
```

Tabela 6.6: Código XML de um ficheiro com dados do ZigBee.

6.2.4 Programação do Circuito

Para efectuar a programação do circuito integrado, é necessário que o módulo se encontre alimentado e que o cabo de comunicação de dados USB se encontre conectado ao computador. Reunidas estas duas condições, é necessário ainda, fazer *reset* ao módulo imediatamente antes de correr o programa *prog.sh*. Para fazer *reset* ao módulo é necessário carregar no respectivo botão SW1 que se encontra na face oposta ao suporte do cartão μSD.

O ficheiro de código que contempla os comandos da Tabela 6.3, necessário para a programação do circuito integrado é o seguinte:

```
#!/bin/bash

echo "Converting elf file to ihex..."
avr32-objcopy -O ihex AudioGuide.elf AudioGuide.hex
```

```
6  echo "Erasing part ($1)..."
   dfu-programmer $1 erase
8
   echo "Flashing part ($1)..."
10 dfu-programmer $1 flash --suppress-bootloader-mem AudioGuide.
      hex

12 echo "Resetting part ($1)..."
   dfu-programmer $1 reset
14
   echo "Starting app..."
16 dfu-programmer $1 start

18 echo "Ready to go!!!"
```

Para a correcta programação do microcontrolador é necessário passar pela linha de comandos o modelo deste, ou seja, o comando a executar é o seguinte:

```
./prog.sh at32uc3b0256
```

Capítulo 7

Manual de Utilizador

As funcionalidades disponibilizadas pelo circuito são muito simples e práticas de aceder.

O botão *SW1*, localizado na face oposta do cartão de memória permite ao utilizador fazer RESET do sistema, caso exista alguma anomalia. O utilizador deverá também efectuar um RESET sempre que desligue o circuito da porta de comunicações de dados USB.

O botão, também na face inferior, mas junto ao canto esquerdo, permite ao utilizador adormecer o sistema para poupança da bateria.

O teclado, permite aceder a mais funcionalidades, nomeadamente:

- Tecla A - voltar a reproduzir o último ficheiro de áudio;
- Tecla B - activar/desactivar o módulo ZigBee;
- Tecla D - activar/desactivar o módulo GPS;
- Tecla 0 - alterar o idioma do sistema.

A alimentação do equipamento está a cargo de 4 baterias AA. E a sua utilização pressupõe a existência de cobertura de rede GPS e/ou ZigBee.

Para adicionar novos dados o utilizador deve substituir o bloco $<!--TheEnd-->$ pelos dados que pretende nos ficheiros XML das Tabelas 6.5 e 6.6. De seguida deve carregar esses dados para o cartão.

Capítulo 8

Resultados

8.1 Consumos Energéticos

Os consumos energéticos medidos, à tensão de 3,3V, estão referenciados na Tabela 8.1.

Elementos Activos	Consumo (mA)
GPS	$\simeq 100$
ZigBee	$\simeq 80$
GPS + ZigBee	$\simeq 130$

Tabela 8.1: Alcance do módulo ZigBee.

8.2 Resultados Práticos

Em ambos os módulos verificou-se que a posição em que o sistema é colocado influencia fortemente as distâncias medidas.

Os obstáculos utilizados nos testes seguintes foram paredes e pessoas, elementos comuns no meio onde se pretende implementar este sistema.

Disposição física	Alcance (m)
Módulo descoberto	6 a 10
Módulo com obstáculos	3 a 5
Módulo rodeado por obstáculos	2

Tabela 8.2: Alcance do módulo ZigBee.

No caso do GPS existem mais factores passiveis de afectarem as medidas, pelo que a incerteza aumenta significativamente. Para além disso as próprias medições efectuadas apresentam um erro associado, por não terem sido rigorosas.

Disposição física	Alcance (m)
Módulo descoberto	> 5
Módulo com obstáculos	> 10
Módulo rodeado por obstáculos	> 20

Tabela 8.3: Alcance do módulo ZigBee.

Capítulo 9

Conclusão e Visão Futura

Em termos gerais, conclui-se que as tecnologias de localização aplicadas no desenvolvimento deste projecto foram escolhas adequadas, pois apresentaram desempenhos bastante satizfatórios, e atingiram os objectivos a que se destinavam.. Enquanto que o GPS apresenta um erro na ordem das dezenas de metros, o ZigBee seria de esperar que melhorasse esses valores para a ordem das unidades, o que de facto se veio a comprovar, como verificam os resultados obtidos.

Os resultados energéticos obtidos são aceitáveis, visto não estarem muito afastados do esperado tendo em consideração os valores fornecidos pelos fabricantes.

Após a montagem do primeiro circuito, há que referir que existiam algumas melhorias a realizar no circuito, por forma a eliminar algumas limitações sérias no circuito. A saber:

- O cartão de cartão de memória deveria ser rodado 180º em torno do seu ponto central, para ficar virado para a extremidade da placa e facilitar o acesso acesso ao mesmo, melhorando significativamente a sua introdução e remoção.

- O suporte de cartão utilizado não possui detecção da presença do cartão pelo que o sistema não tem forma de saber se este está presente ou não. Para um correcto funcionamento o cartão deve ser mantido no suporte sempre que o módulo esteja alimentado, sobre pena dos dados deste serem corrompidos.

- Seriam necessários dois novos reguladores de tensão para garantir um melhor e mais seguro funcionamento do circuito integrado responsável pelo áudio.

- Existem alguns pequenos erros no layout da placa que deveriam ser corrigidos, nomeadamente na memória flash onde existem pinos trocados.

- Para colocar em funcionamento o módulo de FM, seria necessário re-ordenar o bus SPI ou alterar a *framework* da atmel, uma que esta só suporta 4 dispositivos por referência directa. A solução mais simples seria colocar alguma lógica adicional e fazer uso da capacidade de multiplexagem do bus SPI.

- Correcção do *layout* do módulo GPS para o modelo utilizado, que é diferente do que tinha sido idealizado.

- Podem também ser feitas melhorias gerais da placa que permitam diminuir a sua dimensão (exemplo disso seria uma ficha de áudio de menores dimensões), bem como melhorias do software que o tornem mais eficiente do ponto de vista energético.

- A necessidade da montagem do circuito completo na mesma placa, por forma a eliminar alguns erros que surgem nas comunicações USART do sistema que foram efectuadas por meio de fios.

- A introdução de novas funcionalidades é sempre uma opção a ter em conta, de forma a tornar o projecto mais robusto.

O sistema apenas possui dois idiomas (Português e Inglês), mas encontra-se preparado para outros. Para tal, basta que os respectivos dados sejam carregados para o cartão de memória.

Os dados utilizados para o testes foram colocados manualmente no ficheiro XML, pelo que para o utilizador seria preferível uma aplicação para a criação/edição destes ficheiros.

No desenvolvimento da aplicação, surgiram alguns atrasos originados pelo facto de se tratar de uma plataforma de desenvolvimento nova e desconhecida. Estes atrasos provocaram algumas limitações nas funcionalidades do sistema.

Seria também interessante avaliar o comportamento para diferentes níveis de potência dos módulos ZigBee, o que não foi efectuado por já se encontrar um pouco fora do âmbito do projecto.

Tendo em conta as limitações anteriores e a complexidade do sistema, o resultado final pode ser considerado positivo, uma vez que os objectivos principais foram alcançados.

Para além de identificadas as falhas foram também enunciadas meios para as evitar numa próxima revisão do circuito.

Anexos

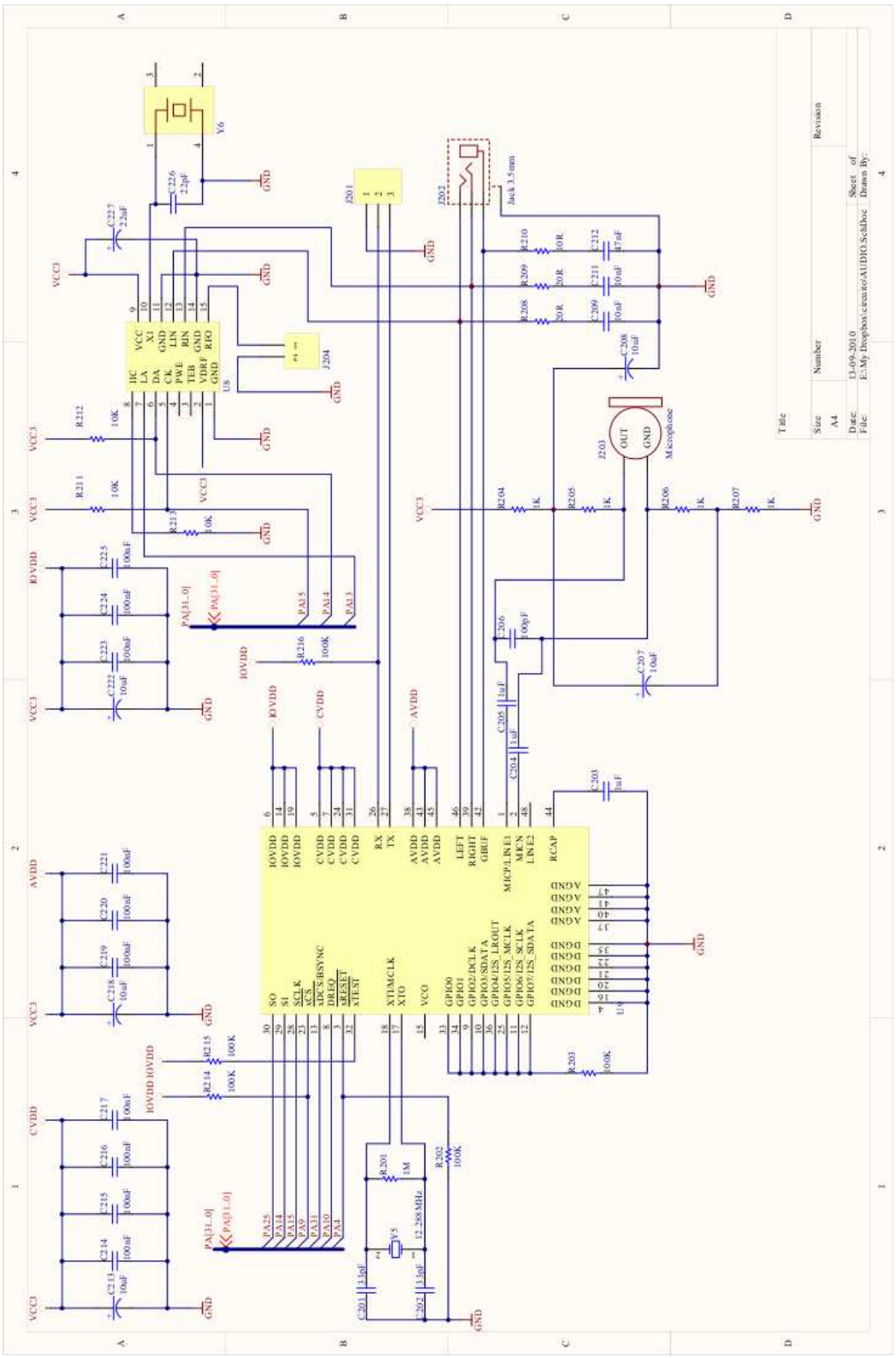


Figura 9.1: Esquema do projecto - áudio.

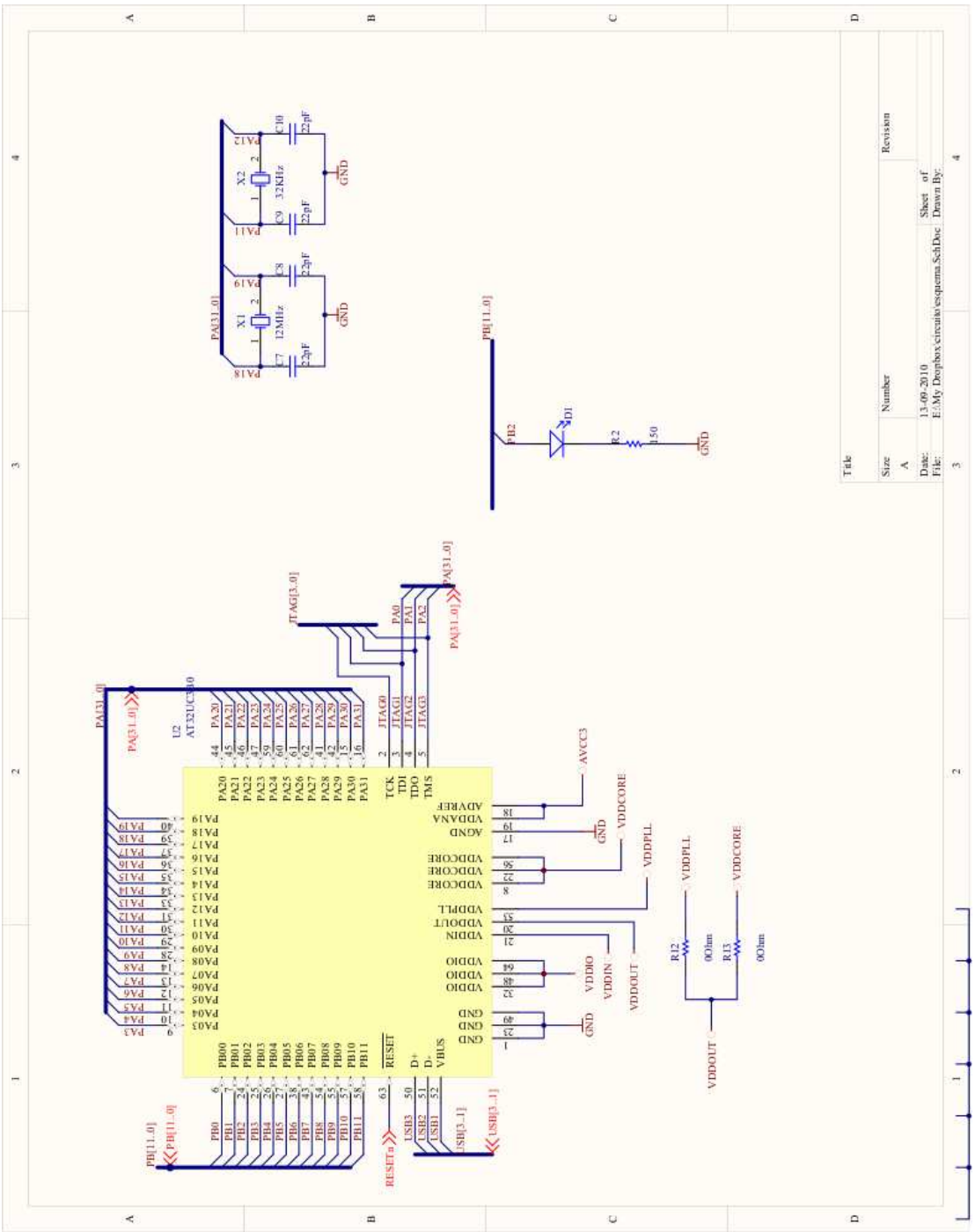


Figura 9.2: Esquema do projecto - microcontrolador.

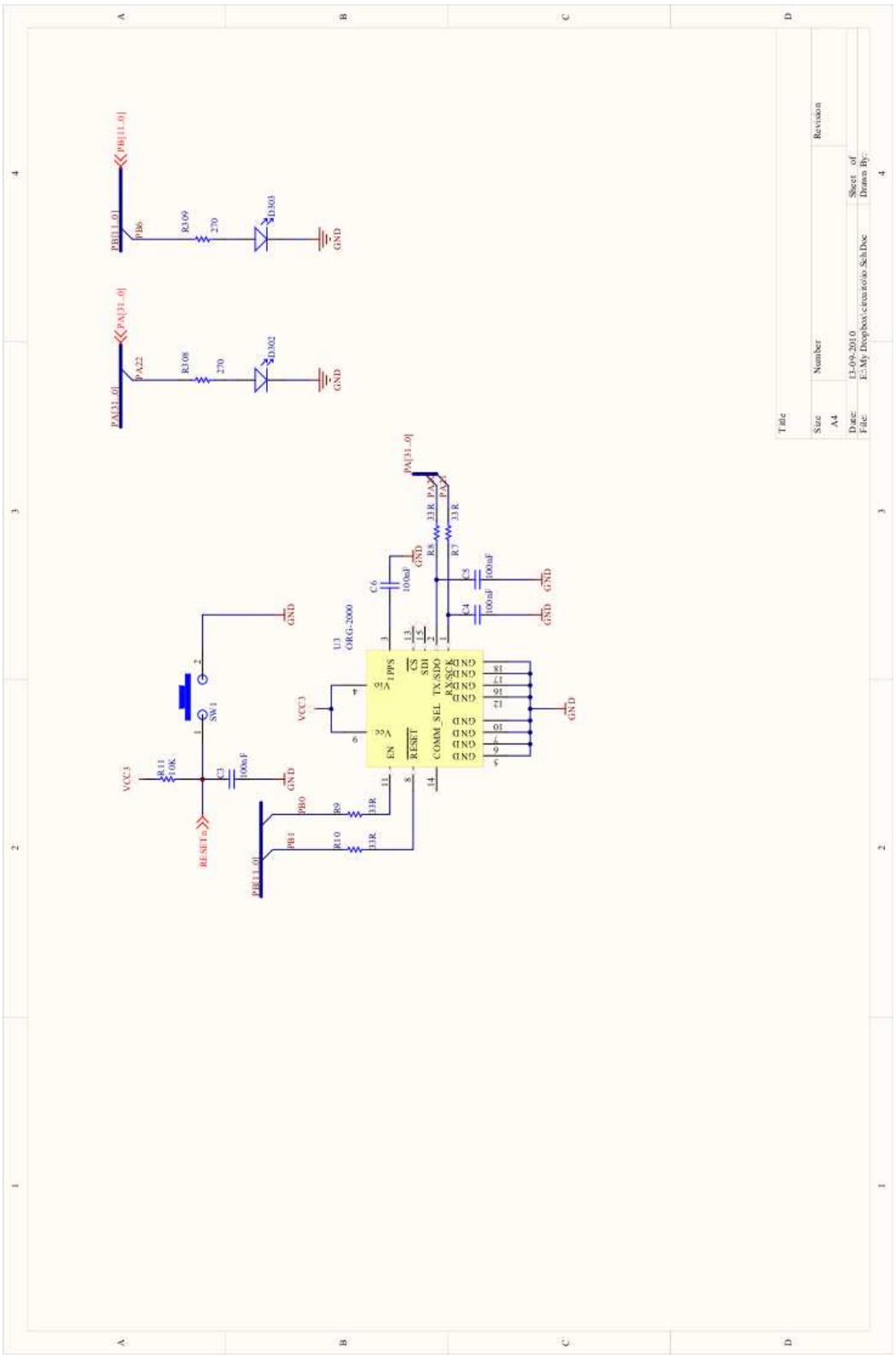


Figura 9.3: Esquema do projecto - GPS.

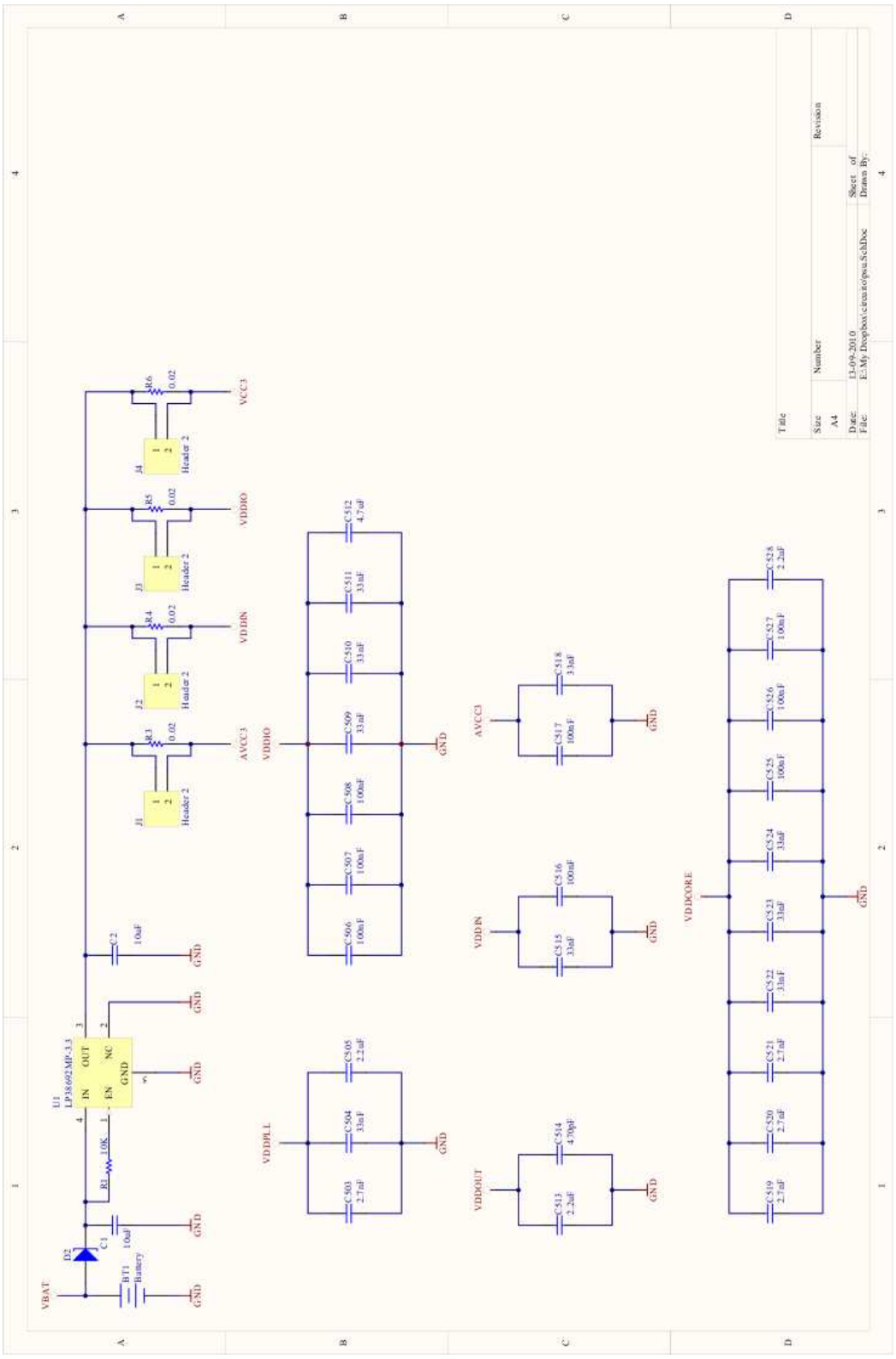


Figura 9.4: Esquema do projecto - fonte de alimentação.

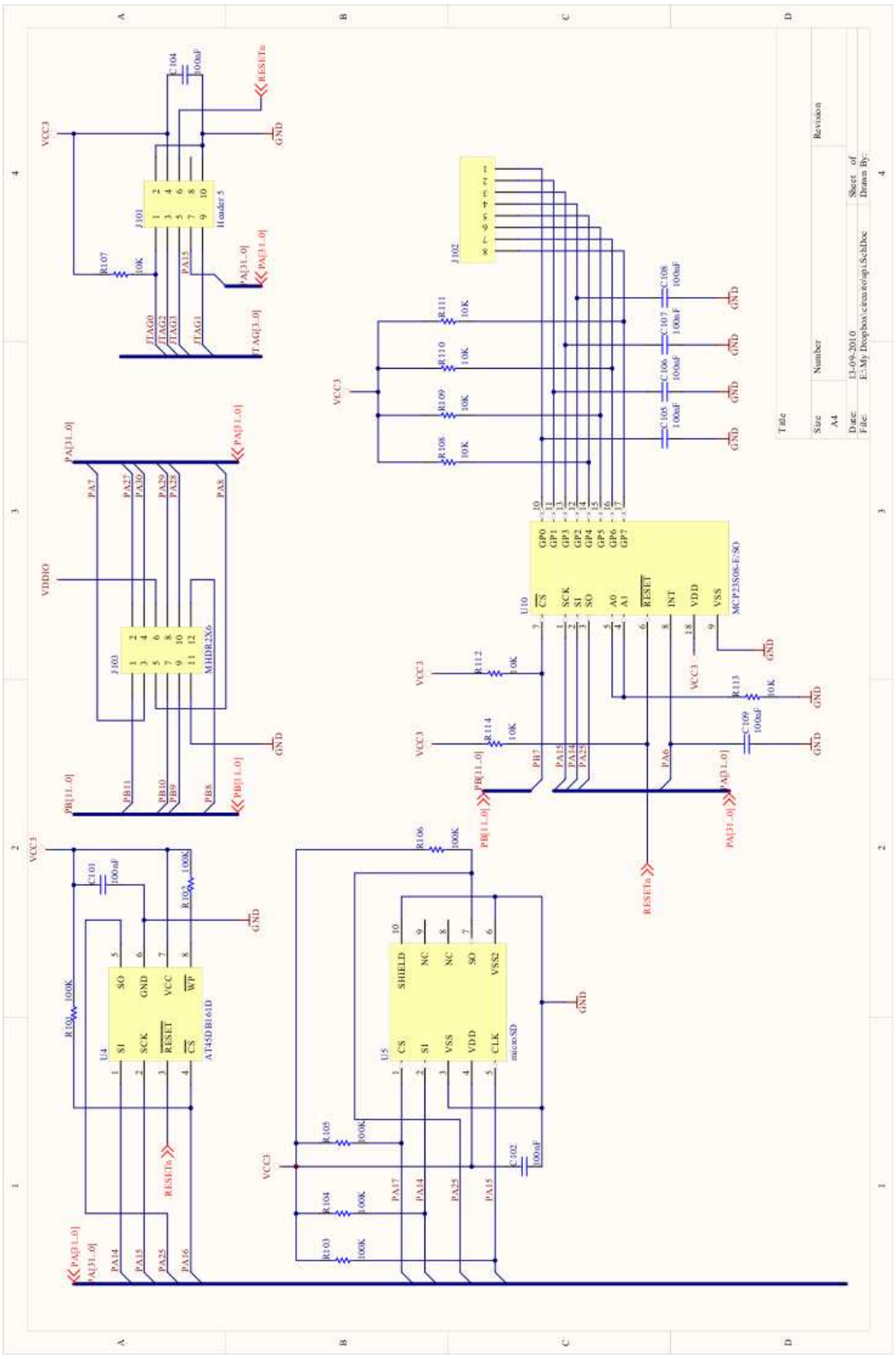


Figura 9.5: Esquema do projecto - dispositivos SPI.

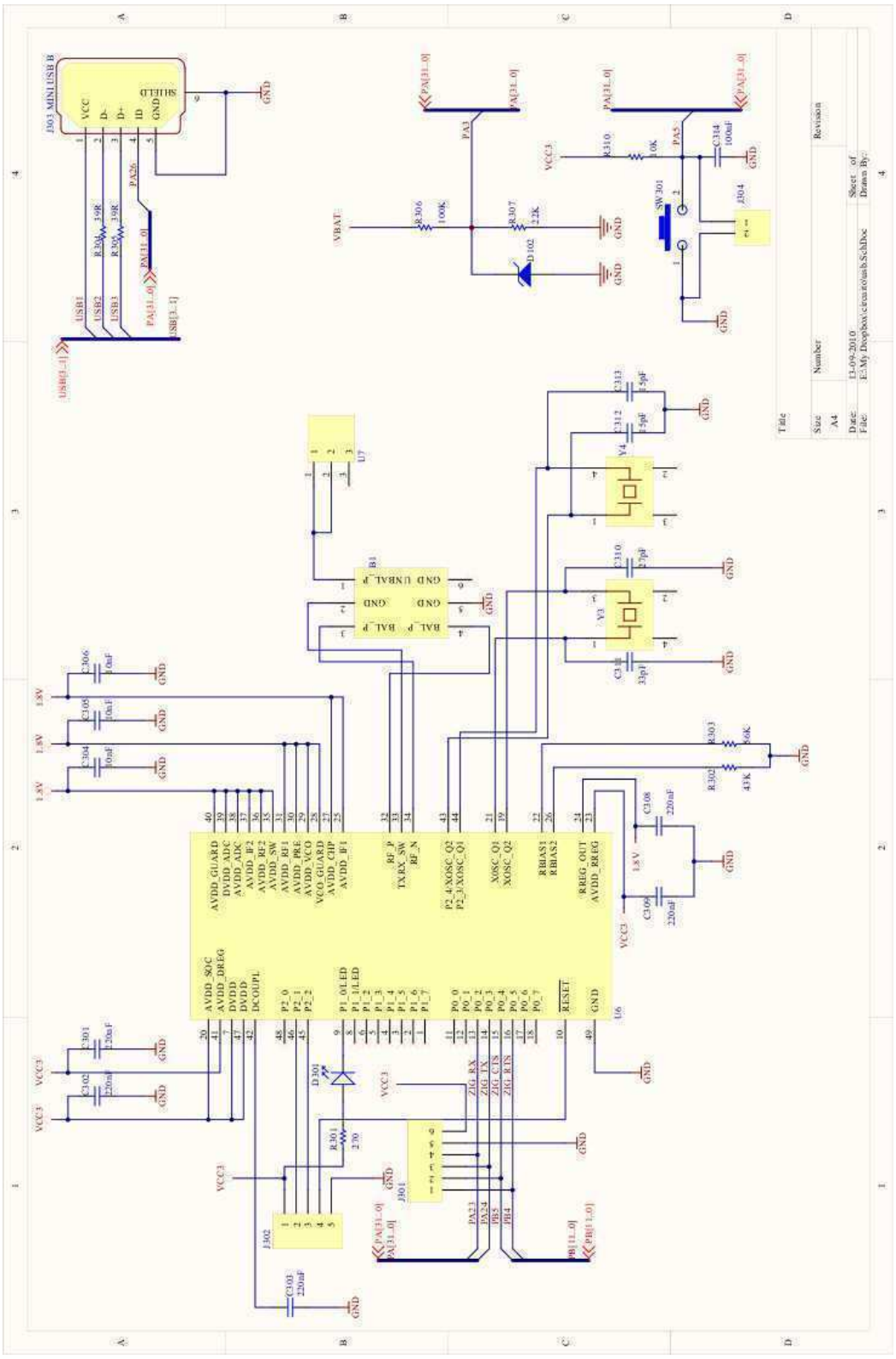


Figura 9.6: Esquema do projecto - ZigBee.

```
/*This file is prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************
 *
 * \brief Main file of the device.
 *
 *
 * \author                Diogo
 *
 ******************************************************************/



//_____  I N C L U D E S ___________________________________________

#ifndef FREERTOS_USED
  #if (defined __GNUC__)
    #include "nlao_cpu.h"
  #endif
#else
  #include <stdio.h>
#endif
#include "compiler.h"
#include "preprocessor.h"
#include "mass_storage_example.h"
#include "board.h"
#include "print_funcs.h"
#include "intc.h"
#include "power_clocks_lib.h"
#include "gpio.h"
#include "ctrl_access.h"
#if (defined AT45DBX_MEM) && (AT45DBX_MEM == ENABLE)
  #include "spi.h"
  #include "conf_at45dbx.h"
#endif
#if (defined SD_MMC_SPI_MEM) && (SD_MMC_SPI_MEM == ENABLE)
  #include "spi.h"
  #include "conf_sd_mmc_spi.h"
#endif
#if ((defined SD_MMC_MCI_0_MEM) && (SD_MMC_MCI_0_MEM == ENABLE)) || ((
```

```
    defined SD_MMC_MCI_1_MEM) && (SD_MMC_MCI_1_MEM == ENABLE))
  #include "mci.h"
41  #include "conf_sd_mmc_mci.h"
#endif
43 #ifdef FREERTOS_USED
  #include "FreeRTOS.h"
45  #include "task.h"
#endif
47 #include "conf_usb.h"
#include "usb_task.h"
49 #if USB_DEVICE_FEATURE == ENABLED
  #include "device_mass_storage_task.h"
51 #endif
#if USB_HOST_FEATURE == ENABLED
53  #include "host_mass_storage_task.h"
#endif
55 #include "ushell_task.h"

57 #include "usart.h"
#include "print_funcs.h"
59


61 //_____ D E F I N I T I O N S
    ---------------------------------------------

63 /*! \name System Clock Frequencies
 */
65 //! @{
static pcl_freq_param_t pcl_freq_param =
67 {
  .cpu_f        = FCPU_HZ,
69  .pba_f        = FPBA_HZ,
  .osc0_f       = FOSC0,
71  .osc0_startup = OSC0_STARTUP
};
73 //! @}

75 /*! \name Hmatrix bus configuration
 */
77 void init_hmatrix(void)
```

```
{
79   union
     {
81     unsigned long                    scfg;
       avr32_hmatrix_scfg_t             SCFG;
83   } u_avr32_hmatrix_scfg;

85   // For the internal-flash HMATRIX slave, use last master as default.
     u_avr32_hmatrix_scfg.scfg = AVR32_HMATRIX.scfg[
         AVR32_HMATRIX_SLAVE_FLASH];
87   u_avr32_hmatrix_scfg.SCFG.defmstr_type =
         AVR32_HMATRIX_DEFMSTR_TYPE_LAST_DEFAULT;
     AVR32_HMATRIX.scfg[AVR32_HMATRIX_SLAVE_FLASH] = u_avr32_hmatrix_scfg.
         scfg;
89 }

91
#ifndef FREERTOS_USED
93
  #if (defined __GNUC__)
95
/*! \brief Low-level initialization routine called during startup, before
       the
97  *        main function.
 *
99  * This version comes in replacement to the default one provided by the
       Newlib
 * add-ons library.
101 * Newlib add-ons' _init_startup only calls init_exceptions, but Newlib
       add-ons'
 * exception and interrupt vectors are defined in the same section and
       Newlib
103 * add-ons' interrupt vectors are not compatible with the interrupt
       management
 * of the INTC module.
105 * More low-level initializations are besides added here.
 */
107 int _init_startup(void)
{
109   // Import the Exception Vector Base Address.
```

```
    extern void _evba;
111
    // Load the Exception Vector Base Address in the corresponding system
        register.
113 Set_system_register(AVR32_EVBA, (int)&_evba);

115 // Enable exceptions.
    Enable_global_exception();
117
    // Initialize interrupt handling.
119 INTC_init_interrupts();

121 // Give the used CPU clock frequency to Newlib, so it can work properly
        .
    set_cpu_hz(pcl_freq_param.pba_f);
123
    // Don't-care value for GCC.
125 return 1;
}
127
  #elif __ICCAVR32__
129
/*! \brief Low-level initialization routine called during startup, before
     the
131  *         main function.
 */
133 int __low_level_init(void)
{
135
#if BOARD == UC3C_EK
137         // Disable WDT At Startup
            AVR32_WDT.ctrl = 0x55000000;
139         AVR32_WDT.ctrl = 0xAA000000;
#endif
141
    // Enable exceptions.
143 Enable_global_exception();

145 // Initialize interrupt handling.
    INTC_init_interrupts();
```

```
147
   // Request initialization of data segments.
149   return 1;
}
151
   #endif  // Compiler
153
#endif  // FREERTOS_USED
155


157 #if (defined AT45DBX_MEM) && (AT45DBX_MEM == ENABLE)

159 /*! \brief Initializes AT45DBX resources: GPIO, SPI and AT45DBX.
 */
161 static void at45dbx_resources_init(void)
{
163   static const gpio_map_t AT45DBX_SPI_GPIO_MAP =
   {
165     {AT45DBX_SPI_SCK_PIN,           AT45DBX_SPI_SCK_FUNCTION          },
        // SPI Clock.
     {AT45DBX_SPI_MISO_PIN,          AT45DBX_SPI_MISO_FUNCTION         },
        // MISO.
167     {AT45DBX_SPI_MOSI_PIN,          AT45DBX_SPI_MOSI_FUNCTION         },
        // MOSI.
#define AT45DBX_ENABLE_NPCS_PIN(NPCS, unused) \
169     {AT45DBX_SPI_NPCS##NPCS##_PIN, AT45DBX_SPI_NPCS##NPCS##_FUNCTION},
        // Chip Select NPCS.
    MREPEAT(AT45DBX_MEM_CNT, AT45DBX_ENABLE_NPCS_PIN, ~)
171 #undef AT45DBX_ENABLE_NPCS_PIN
   };
173
   // SPI options.
175   spi_options_t spiOptions =
   {
177     .reg          = AT45DBX_SPI_FIRST_NPCS,   // Defined in conf_at45dbx.
        h.
     .baudrate     = AT45DBX_SPI_MASTER_SPEED, // Defined in conf_at45dbx.
        h.
179     .bits         = AT45DBX_SPI_BITS,         // Defined in conf_at45dbx.
        h.
```

```
    .spck_delay   = 0,
    .trans_delay  = 0,
    .stay_act     = 1,
    .spi_mode     = 0,
    .modfdis      = 1
  };

  // Assign I/Os to SPI.
  gpio_enable_module(AT45DBX_SPI_GPIO_MAP,
                     sizeof(AT45DBX_SPI_GPIO_MAP) / sizeof(
                         AT45DBX_SPI_GPIO_MAP[0]));

  // If the SPI used by the AT45DBX is not enabled.
  if (!spi_is_enabled(AT45DBX_SPI))
  {
    // Initialize as master.
    spi_initMaster(AT45DBX_SPI, &spiOptions);

    // Set selection mode: variable_ps, pcs_decode, delay.
    spi_selectionMode(AT45DBX_SPI, 0, 0, 0);

    // Enable SPI.
    spi_enable(AT45DBX_SPI);
  }

  // Initialize data flash with SPI PB clock.
  at45dbx_init(spiOptions, pcl_freq_param.pba_f);
}

#endif  // AT45DBX_MEM == ENABLE

#if (defined SD_MMC_SPI_MEM) && (SD_MMC_SPI_MEM == ENABLE)

/*! \brief Initializes SD/MMC resources: GPIO, SPI and SD/MMC.
 */
static void sd_mmc_spi_resources_init(void)
{
  static const gpio_map_t SD_MMC_SPI_GPIO_MAP =
  {
    {SD_MMC_SPI_SCK_PIN,  SD_MMC_SPI_SCK_FUNCTION },  // SPI Clock.
```

```
219     {SD_MMC_SPI_MISO_PIN, SD_MMC_SPI_MISO_FUNCTION},   // MISO.
        {SD_MMC_SPI_MOSI_PIN, SD_MMC_SPI_MOSI_FUNCTION},   // MOSI.
221     {SD_MMC_SPI_NPCS_PIN, SD_MMC_SPI_NPCS_FUNCTION}    // Chip Select NPCS
            .
    };
223
    // SPI options.
225 spi_options_t spiOptions =
    {
227   .reg          = SD_MMC_SPI_NPCS,
      .baudrate     = SD_MMC_SPI_MASTER_SPEED,   // Defined in
          conf_sd_mmc_spi.h.
229   .bits         = SD_MMC_SPI_BITS,           // Defined in
          conf_sd_mmc_spi.h.
      .spck_delay   = 0,
231   .trans_delay  = 0,
      .stay_act     = 1,
233   .spi_mode     = 0,
      .modfdis      = 1
235 };

237 // Assign I/Os to SPI.
    gpio_enable_module(SD_MMC_SPI_GPIO_MAP,
239                    sizeof(SD_MMC_SPI_GPIO_MAP) / sizeof(
                           SD_MMC_SPI_GPIO_MAP[0]));

241 // If the SPI used by the SD/MMC is not enabled.
    if (!spi_is_enabled(SD_MMC_SPI))
243 {
      // Initialize as master.
245   spi_initMaster(SD_MMC_SPI, &spiOptions);

247   // Set selection mode: variable_ps, pcs_decode, delay.
      spi_selectionMode(SD_MMC_SPI, 0, 0, 0);
249
      // Enable SPI.
251   spi_enable(SD_MMC_SPI);
    }
253
    // Initialize SD/MMC with SPI PB clock.
```

```
255   sd_mmc_spi_init(spiOptions, pcl_freq_param.pba_f);
}
257
#endif  // SD_MMC_SPI_MEM == ENABLE
259
261 #if ((defined SD_MMC_MCI_0_MEM) && (SD_MMC_MCI_0_MEM == ENABLE)) || ((
        defined SD_MMC_MCI_1_MEM) && (SD_MMC_MCI_1_MEM == ENABLE))
263 /*! \brief Initializes SD/MMC resources: GPIO, MCI and SD/MMC.
 */
265 static void sd_mmc_mci_resources_init(void)
{
267   static const gpio_map_t SD_MMC_MCI_GPIO_MAP =
   {
269     {SD_SLOT_8BITS_CLK_PIN,    SD_SLOT_8BITS_CLK_FUNCTION   }, // SD CLK.
     {SD_SLOT_8BITS_CMD_PIN,    SD_SLOT_8BITS_CMD_FUNCTION   }, // SD CMD.
271     {SD_SLOT_8BITS_DATA0_PIN, SD_SLOT_8BITS_DATA0_FUNCTION}, // SD DAT
         [0].
     {SD_SLOT_8BITS_DATA1_PIN, SD_SLOT_8BITS_DATA1_FUNCTION}, // DATA Pin
         .
273     {SD_SLOT_8BITS_DATA2_PIN, SD_SLOT_8BITS_DATA2_FUNCTION}, // DATA Pin
         .
     {SD_SLOT_8BITS_DATA3_PIN, SD_SLOT_8BITS_DATA3_FUNCTION}, // DATA Pin
         .
275     {SD_SLOT_8BITS_DATA4_PIN, SD_SLOT_8BITS_DATA4_FUNCTION}, // DATA Pin
         .
     {SD_SLOT_8BITS_DATA5_PIN, SD_SLOT_8BITS_DATA5_FUNCTION}, // DATA Pin
         .
277     {SD_SLOT_8BITS_DATA6_PIN, SD_SLOT_8BITS_DATA6_FUNCTION}, // DATA Pin
         .
     {SD_SLOT_8BITS_DATA7_PIN, SD_SLOT_8BITS_DATA7_FUNCTION}   // DATA Pin
         .
279   };

281   // MCI options.
   static const mci_options_t MCI_OPTIONS =
283   {
     .card_speed = 400000,
285     .card_slot  = SD_SLOT_8BITS, // Default card initialization.
```

```
  };

  // Assign I/Os to MCI.
  gpio_enable_module(SD_MMC_MCI_GPIO_MAP,
                     sizeof(SD_MMC_MCI_GPIO_MAP) / sizeof(
                         SD_MMC_MCI_GPIO_MAP[0]));

  // Enable pull-up for Card Detect.
  gpio_enable_pin_pull_up(SD_SLOT_8BITS_CARD_DETECT);

  // Enable pull-up for Write Protect.
  gpio_enable_pin_pull_up(SD_SLOT_8BITS_WRITE_PROTECT);

  // Initialize SD/MMC with MCI PB clock.
  sd_mmc_mci_init(&MCI_OPTIONS, pcl_freq_param.pba_f, pcl_freq_param.
      cpu_f);
}

#endif  // SD_MMC_MCI_0_MEM == ENABLE || SD_MMC_MCI_1_MEM == ENABLE


static const usart_options_t DBG_USART_OPTIONS = {
                .baudrate     = 57600,
                .charlength   = 8,
                .paritytype = USART_NO_PARITY,
                .stopbits     = USART_1_STOPBIT,
                .channelmode = USART_NORMAL_CHMODE
};


/*! \brief Main function. Execution starts here.
 *
 * \retval 42 Fatal error.
 */
int main(void)
{
  // Configure system clocks.
  if (pcl_configure_clocks(&pcl_freq_param) != PASS)
    return 42;

```

```
  // Initialize USART link.
325  //init_dbg_rs232(pcl_freq_param.pba_f);
  usart_init_rs232(&AVR32_USART1, &DBG_USART_OPTIONS, pcl_freq_param.
      pba_f);
327  usart_write_line(&AVR32_USART1, "AudioGuide v0.1\n");

329 #if (defined AT45DBX_MEM) && (AT45DBX_MEM == ENABLE)
  at45dbx_resources_init();
331 #endif

333 #if SD_MMC_SPI_MEM == ENABLE
  sd_mmc_spi_resources_init();
335 #endif

337 #if ((defined SD_MMC_MCI_0_MEM) && (SD_MMC_MCI_0_MEM == ENABLE)) || ((
    defined SD_MMC_MCI_1_MEM) && (SD_MMC_MCI_1_MEM == ENABLE))
  sd_mmc_mci_resources_init();
339 #endif

341 #ifdef FREERTOS_USED
  if (!ctrl_access_init())
343  {
    portDBG_TRACE("The module CTRL_ACCESS could not be initialized.");
345    return 42;
  }
347 #endif  // FREERTOS_USED

349  // Init Hmatrix bus
  init_hmatrix();
351
  // Initialize USB clock.
353  pcl_configure_usb_clock();

355  // Initialize USB tasks.
  usb_task_init();
357 #if USB_DEVICE_FEATURE == ENABLED
  device_mass_storage_task_init();
359 #endif
 #if USB_HOST_FEATURE == ENABLED
361  host_mass_storage_task_init();
```

```
#endif
363   ushell_task_init(pcl_freq_param.pba_f);

365 #ifdef FREERTOS_USED
   vTaskStartScheduler();
367   portDBG_TRACE("FreeRTOS returned.");
   return 42;
369 #else
   while (TRUE)
371   {
     usb_task();
373   #if USB_DEVICE_FEATURE == ENABLED
     device_mass_storage_task();
375   #endif
   #if USB_HOST_FEATURE == ENABLED
377     host_mass_storage_task();
   #endif
379     ushell_task();
   }
381 #endif  // FREERTOS_USED
}
383

static const gpio_map_t USART_GPIO_MAP = {                // USART pins
385         {AVR32_USART1_RXD_0_0_PIN, AVR32_USART1_RXD_0_0_FUNCTION},
         {AVR32_USART1_TXD_0_0_PIN, AVR32_USART1_TXD_0_0_FUNCTION}
387 };

389 static const usart_options_t USART_OPTIONS = {  // USART options
         .baudrate     = 57600,
391         .charlength   = 8,
         .paritytype   = USART_NO_PARITY,
393         .stopbits     = USART_1_STOPBIT,
         .channelmode  = USART_NORMAL_CHMODE
395 };

397 static const gpio_map_t USART_GPS_GPIO_MAP = {            // USART pins
         {AVR32_USART2_RXD_0_1_PIN, AVR32_USART2_RXD_0_1_FUNCTION},
399         {AVR32_USART2_TXD_0_1_PIN, AVR32_USART2_TXD_0_1_FUNCTION}
};
401
```

```
static const usart_options_t USART_GPS_OPTIONS = {      // USART options
        .baudrate     = 4800,
        .charlength   = 8,
        .paritytype   = USART_NO_PARITY,
        .stopbits     = USART_1_STOPBIT,
        .channelmode  = USART_NORMAL_CHMODE
};

float lat=0.0, lon=0.0;
float lat_old=0.0, lon_old=0.0;
int id = 0;
int id_old = 0;
char *file;

int main (void) {

        float lat=0, lon=0;

        // Configure system clocks.
        if (pcl_configure_clocks(&pcl_freq_param) != PASS) return 42;

        // Configure USARTs
        gpio_enable_module(USART_GPIO_MAP,      // Assign USART I/O
                        sizeof(USART_GPIO_MAP) / sizeof(USART_GPIO_MAP[0]));

        gpio_enable_module(USART_GPS_GPIO_MAP,  // Assign USART I/O
                                sizeof(USART_GPS_GPIO_MAP) / sizeof(
                                        USART_GPS_GPIO_MAP[0]));

        usart_init_rs232(&AVR32_USART1, &USART_OPTIONS, FOSC0);
        usart_init_rs232(&AVR32_USART2, &USART_GPS_OPTIONS, FOSC0);

        // ZigBee data verification
        id = zigbee_get_id()
        while (id != id_old ) {
                if(find_file_gps()) {
                        play(&file);
                }
                id_old = id;
                id = zigbee_get_id();
```

```
441         }

443         // GPS data verification
            get_gps_data();
445
            while ((lat!=lat_old) || (lon!=lon_old)) {
447             if(find_file_gps()) {
                        play(&file);
449             }
                lat_old = lat;
451             lon_old = lon;
                get_gps_data();
453         }

455         // Setting timer to check gps data
            timer0_set(2, FOSC32);
457
            #if (defined AT45DBX_MEM) && (AT45DBX_MEM == ENABLE)
459           at45dbx_resources_init();
            #endif
461
            #if SD_MMC_SPI_MEM == ENABLE
463           sd_mmc_spi_resources_init();
            #endif
465
            #if ((defined SD_MMC_MCI_0_MEM) && (SD_MMC_MCI_0_MEM == ENABLE))
                || ((defined SD_MMC_MCI_1_MEM) && (SD_MMC_MCI_1_MEM == ENABLE)
                )
467           sd_mmc_mci_resources_init();
            #endif
469
            #if USB_MODE_START == ENABLED
471           // Initialize USB clock.
              pcl_configure_usb_clock();
473           // Initialize USB tasks.
              usb_task_init();
475         #if USB_DEVICE_FEATURE == ENABLED
              device_mass_storage_task_init();
477         #endif
```

```
479         ushell_task_init(pcl_freq_param.pba_f);

481         #ifdef FREERTOS_USED
              vTaskStartScheduler();
483           portDBG_TRACE("FreeRTOS returned.");
              return 42;
485         #else
              while (TRUE)
487           {
                usb_task();
489           #if USB_DEVICE_FEATURE == ENABLED
                device_mass_storage_task();
491           #endif
              #if USB_HOST_FEATURE == ENABLED
493             host_mass_storage_task();
              #endif
495             ushell_task();
              }
497         #endif  // FREERTOS_USED
            #else
499           while(1);
            #endif
501         return 0;
    }
```

Definições e serviços do circuito.

```
/*This file is prepared for Doxygen automatic documentation generation.*/
/*! \file
    ******************************************************************
 *
 * \brief Board header file.
 *
 * This file contains definitions and services related to the features of
     the
 * proto board
 *
 * To use this board, define BOARD=EVK1101.
 *
 * - Compiler:           IAR EWAVR32 and GNU GCC for AVR32
 * - Supported devices:  All AVR32 AT32UC3B devices can be used.
 * - AppNote:
 *
 * \author               Diogo
 *
 ******************************************************************
    */

#ifndef _EVK1101_H_
#define _EVK1101_H_

#include "compiler.h"

#ifdef __AVR32_ABI_COMPILER__ // Automatically defined when compiling for
    AVR32, not when assembling.
#  include "led.h"
#endif  // __AVR32_ABI_COMPILER__


/*! \name Oscillator Definitions
 */
//! @{

// RCOsc has no custom calibration by default. Set the following
    definition to
```

```
// the appropriate value if a custom RCOsc calibration has been applied to your
// part.
//#define FRCOSC            AVR32_PM_RCOSC_FREQUENCY                //!< RCOsc frequency: Hz.

#define FOSC32            32000                                   //!< Osc32 frequency: Hz.
#define OSC32_STARTUP     AVR32_PM_OSCCTRL32_STARTUP_8192_RCOSC   //!< Osc32 startup time: RCOsc periods.

#define FOSC0             12000000                                //!< Osc0 frequency: Hz.
#define OSC0_STARTUP      AVR32_PM_OSCCTRL0_STARTUP_2048_RCOSC    //!< Osc0 startup time: RCOsc periods.

// Osc1 crystal is not mounted by default. Set the following definitions to the
// appropriate values if a custom Osc1 crystal is mounted on your board.
//#define FOSC1             12000000                                //!< Osc1 frequency: Hz.
//#define OSC1_STARTUP      AVR32_PM_OSCCTRL1_STARTUP_2048_RCOSC    //!< Osc1 startup time: RCOsc periods.

//! @}


/*! \name USB Definitions
 */
//! @{

//! Multiplexed pin used for USB_ID: AVR32_USBB_USB_ID_x_x.
//! To be selected according to the AVR32_USBB_USB_ID_x_x_PIN and
//! AVR32_USBB_USB_ID_x_x_FUNCTION definitions from <avr32/uc3bxxxx.h>.
#define USB_ID                            AVR32_USBB_USB_ID_0_0

//! Multiplexed pin used for USB_VBOF: AVR32_USBB_USB_VBOF_x_x.
//! To be selected according to the AVR32_USBB_USB_VBOF_x_x_PIN and
//! AVR32_USBB_USB_VBOF_x_x_FUNCTION definitions from <avr32/uc3bxxxx.h>.
#define USB_VBOF                          AVR32_USBB_USB_VBOF_0_0
```

```
65
   //! Active level of the USB_VBOF output pin.
67 #define USB_VBOF_ACTIVE_LEVEL          LOW

69 //! USB overcurrent detection pin.
   #define USB_OVERCURRENT_DETECT_PIN   AVR32_PIN_PA20
71
   //! @}
73


75 //! Number of LEDs.
   #define LED_COUNT    4
77
   /*! \name GPIO Connections of LEDs
79  */
   //! @{
81 #define LED0_GPIO    AVR32_PIN_PA22
   #define LED1_GPIO    AVR32_PIN_PB06
83 #define LED2_GPIO    AVR32_PIN_PB02
   //! @}
85
   /*! \name Color Identifiers of LEDs to Use with LED Functions
87  */
   //! @{
89 #define LED_MONO0_GREEN    LED0
   #define LED_MONO1_GREEN    LED1
91 #define LED_MONO2_GREEN    LED2
   #define LED_MONO3_GREEN    LED3
93 //! @}


95
   /*! \name GPIO Connections of Push Button and Keypad
97  */
   //! @{
99 #define GPIO_PUSH_BUTTON                          AVR32_PIN_PA05
   #define GPIO_PUSH_BUTTON                          0
101#define GPIO_KEYPAD_INT                           AVR32_PIN_PA06
   #define GPIO_KEYPAD_INT_PRESSED          0
103//! @}
```

```
105
/*! \name ADC Connection of the Battery
107  */
//! @{
109 #define ADC_BATTERY_CHANNEL       0
#define ADC_BATTERY_PIN           AVR32_ADC_AD_0_PIN
111 #define ADC_BATTERY_FUNCTION      AVR32_ADC_AD_0_FUNCTION
//! @}
113



115


117 /*! \name SPI Connections of the AT45DBX Data Flash Memory
 */
119 //! @{
#define AT45DBX_SPI                   (&AVR32_SPI)
121 #define AT45DBX_SPI_NPCS              0
#define AT45DBX_SPI_SCK_PIN           AVR32_SPI_SCK_0_0_PIN
123 #define AT45DBX_SPI_SCK_FUNCTION      AVR32_SPI_SCK_0_0_FUNCTION
#define AT45DBX_SPI_MISO_PIN          AVR32_SPI_MISO_0_0_PIN
125 #define AT45DBX_SPI_MISO_FUNCTION     AVR32_SPI_MISO_0_0_FUNCTION
#define AT45DBX_SPI_MOSI_PIN          AVR32_SPI_MOSI_0_0_PIN
127 #define AT45DBX_SPI_MOSI_FUNCTION     AVR32_SPI_MOSI_0_0_FUNCTION
#define AT45DBX_SPI_NPCS0_PIN         AVR32_SPI_NPCS_0_0_PIN
129 #define AT45DBX_SPI_NPCS0_FUNCTION    AVR32_SPI_NPCS_0_0_FUNCTION
//! @}
131


133 /*! \name GPIO and SPI Connections of the SD/MMC Connector
 */
135 //! @{
//#define SD_MMC_CARD_DETECT_PIN        AVR32_PIN_PB00
137 #define SD_MMC_WRITE_PROTECT_PIN      AVR32_PIN_PB01
#define SD_MMC_SPI                    (&AVR32_SPI)
139 #define SD_MMC_SPI_NPCS               1
#define SD_MMC_SPI_SCK_PIN            AVR32_SPI_SCK_0_0_PIN
141 #define SD_MMC_SPI_SCK_FUNCTION       AVR32_SPI_SCK_0_0_FUNCTION
#define SD_MMC_SPI_MISO_PIN           AVR32_SPI_MISO_0_0_PIN
143 #define SD_MMC_SPI_MISO_FUNCTION      AVR32_SPI_MISO_0_0_FUNCTION
#define SD_MMC_SPI_MOSI_PIN           AVR32_SPI_MOSI_0_0_PIN
```

```
145 #define SD_MMC_SPI_MOSI_FUNCTION      AVR32_SPI_MOSI_0_0_FUNCTION
    #define SD_MMC_SPI_NPCS_PIN           AVR32_SPI_NPCS_1_0_PIN
147 #define SD_MMC_SPI_NPCS_FUNCTION      AVR32_SPI_NPCS_1_0_FUNCTION
    //! @}
149
151
    /*! \name SPI Connections of the Spare SPI Connector
153  */
    //! @{
155 #define SPARE_SPI                     (&AVR32_SPI)
    #define SPARE_SPI_NPCS                2
157 #define SPARE_SPI_SCK_PIN             AVR32_SPI_SCK_0_0_PIN
    #define SPARE_SPI_SCK_FUNCTION        AVR32_SPI_SCK_0_0_FUNCTION
159 #define SPARE_SPI_MISO_PIN            AVR32_SPI_MISO_0_0_PIN
    #define SPARE_SPI_MISO_FUNCTION       AVR32_SPI_MISO_0_0_FUNCTION
161 #define SPARE_SPI_MOSI_PIN            AVR32_SPI_MOSI_0_0_PIN
    #define SPARE_SPI_MOSI_FUNCTION       AVR32_SPI_MOSI_0_0_FUNCTION
163 #define SPARE_SPI_NPCS_PIN            AVR32_SPI_NPCS_2_0_PIN
    #define SPARE_SPI_NPCS_FUNCTION       AVR32_SPI_NPCS_2_0_FUNCTION
165 //! @}
167
    #endif  // _EVK1101_H_
```

Definições e serviços do circuito.

```
/*
 * Interrupt Routines
 * interrupts.c
 *
 *  Created on: Oct 27, 2010
 *      Author: Diogo
 */

#include "interrupts.h"
#include "gpio.h"
#include "board.h"
#include "print_funcs.h"
#include "usart.h"
#include <avr32/io.h>

void Enable_Usart_Interrupts (volatile avr32_usart_t *usart, int enableTX
    , int enableRX) {
        if (enableRX && enableTX) {
                usart->ier = 0x00000003;
                return;
        }

        if (enableRX) {
                usart->ier = 0x00000001;
                return;
        }

        if (enableTX) {
                usart->ier = 0x00000002;
                return;
        }

        if (!enableTX && !enableRX) {
                usart->ier = 0x00000000;
                return;
        }
}
```

```
38 __attribute__((__interrupt__)) void buttons_int_handler(void) {
        if(gpio_get_pin_interrupt_flag(GPIO_KEYPAD_INT)) {
40              read_keypad();
                gpio_clear_pin_interrupt_flag(GPIO_KEYPAD_INT);
42      }
}
44
__attribute__((__interrupt__)) void zig_usart_int_handler(void) {
46      if(gpio_get_pin_interrupt_flag(GPIO_USART1_INT)) {
                    id = zigbee_get_id();
48                  gpio_clear_pin_interrupt_flag(GPIO_USART1_INT);
        }
50
}
52
__attribute__((__interrupt__)) void push_buttons_int_handler(void) {
54      if(gpio_get_pin_interrupt_flag(GPIO_PUSH_BUTTON)) {
                hibernate();
56              gpio_clear_pin_interrupt_flag(GPIO_PUSH_BUTTON);
        }
58 }
```

Bibliografia

[1] Garmin Ltd. POI Loader Service. Website, October 2009. `http://www.garmin.com/products/poiloader/`.

[2] D. Estrin N. Bulusu, J. Heidemann. Gps-less low cost outdoor localization for very small devices. *IEEE PERSONAL COMMUNICATIONS MAGAZINE*, October 2000.

[3] V. Johnson L. Reichert T. Strothotte, H. Petrie. Mobic: user needs and preliminary design for a mobility aid for blind and elderly travelleres. *Journal of Navigation*, 49:45–52, 1996.

[4] D. A. Ross and B. B. Blasch. Wearable interfaces for orientation and wayfinding. *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 193–200, 2000.

[5] A. Polychronopoulos, M. Tsogas, A. Amditis, U. Scheunert, L. Andreone, and F. Tango. Dynamic situation and threat assessment for collision waring systemas: the euclide approach. *2004 IEEE Intelligent Vehicles Symposium*, June 2004.

[6] S. Willis and S. Helal. Rfid information grid for blind navigation and wayfinding. *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 34–37, October 2005.

[7] J. Scott M. Hazas and J. Krumm. Proceedings of the 2003 workshop on location-aware computing. *Ubicomp 2003*, October 2003. `http://www.ubicomp.org/ubicomp2003/workshops/locationaware/`.

[8] V. Le T. Nguyen, N. Oh and S. Lee. A low-power cmos direct conversion receiver with 3-db nf and 30-khz flicker-noise corner for 915-mhz band ieee 802.15.4 zigbee

standard. *Microwave Theory and Techniques, IEEE Transactions on*, pages 735–741, February 2006.

[9] E.D. Kaplan and C.J. Hegarty. *Understanding GPS: principles and applications*. Artech House mobile communications series. Artech House, 2006.

[10] I.S. Burnett. *The MPEG-21 book*. Wiley, 2006.

[11] S. Barrett and M. Thornton. *Arduino Microcontroller: Processing for Everyone!* Synthesis Lectures on Digital Circuits and Systems Series. Morgan & Claypool Publishers, 2010.

[12] J. Axelson. *Serial port complete: programming and circuits for RS-232 and RS-485 links and networks*. Lakeview Research, 1998.

[13] J. Catsoulis. *Designing embedded hardware*. O'Reilly Series. O'Reilly, 2005.

[14] P.P. Editor. *1394 Monthly Newsletter*. Information Gatekeepers, Inc.

[15] Atmel. At32uc3b series preliminary. 2010. `http://www.atmel.com/dyn/resources/prod_documents/doc32059.pdf`.

[16] Atmel. At32uc3b series preliminary summary. 2010. `http://www.atmel.com/dyn/resources/prod_documents/32059S.pdf`.

[17] Atmel. Avr32 architecture manual. 2010. `http://www.atmel.com/dyn/resources/prod_documents/doc32000.pdf`.

[18] Atmel. Avr32uc technical reference manual. 2010. `http://www.atmel.com/dyn/resources/prod_documents/doc32059.pdf`.

[19] Origin GPS. Origin gps, 2010. `http://www.origingps.com/product.aspx?pid=25`.

[20] CSR. Sirfstariii gsc3e/lpx. Website, 2010. `http://www.csr.com/products/27/sirfstariii-gsc3elpx`.

[21] Origin GPS. Org-13xx series datasheet. 2009. `http://www.roundsolutions.com/techdocs/gps_antennas/ORG13XX%20Series%20Datasheet.pdf`.

[22] LTD. Niigata Seimitsu Co. Fm transmitter module - description. 2006. http://www.sparkfun.com/datasheets/Wireless/General/NS73_Datasheet.pdf.

[23] LTD. Niigata Seimitsu Co. Ns73 - product specification. 2006. http://www.sparkfun.com/datasheets/Wireless/General/NS73_Description.pdf.

[24] Sparkfun. Fm radio transmitter module - ns73m. 2010. http://www.sparkfun.com/commerce/product_info.php?products_id=8452.

[25] Altium. Altium designer. Website, 2010. http://www.altium.com/altiumsite/products/altium-designer/en/altium-designer_home.cfm.

[26] Euro Circuits. Euro circuits - pcb prototypes. Website, 2010. http://www.eurocircuits.com/.

[27] Euro Circuits. Fast guide to prices and ordering. May 2010. http://www.eurocircuits.com/images/stories/ec09/ec-fast-guide-english-4-2010-v1.pdf.

[28] Ucamco NV. The rs-274x format. *DMS Reference US-GXXX-TB-01-EN-D*, December 2010.

[29] C. Schroeder. *Printed circuit board design using AutoCAD*. EDN series for design engineers. Newnes, 1998.

[30] K.N. King. *C programming: a modern approach*. W.W. Norton & Company, 2008.

[31] Eclipse. Eclipse ide. Website, 2010. http://www.eclipse.org/.

[32] Gnu. Gnu compiler colection - gcc. Website, 2010. http://gcc.gnu.org.

[33] Atmel. Avr32 studio. Website, 2010. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4116.

[34] Atmel. Avr uc3 software framework. Website, February 2010. http://atmel.com/dyn/products/tools_card.asp?tool_id=4192&source=apps-home_appliances-home.

[35] Weston T. Schmidt. dfu-programmer. Website, August 2009. http://dfu-programmer.sourceforge.net/.